IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2021, accepted December 2, 2021, date of publication December 6, 2021,
date of current version December 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3133495

SEQ2SEQ++: A Multitasking-Based Seq2seq Model to Generate Meaningful and Relevant Answers

KULOTHUNKAN PALASUNDRAM, NURFADHLINA MOHD SHAREF, KHAIRUL AZHAR KASMIRAN, AND AZREEN AZMAN, (Member, IEEE)
Intelligent Computing Research Group, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Seri Kembangan, Selangor 43400, Malaysia

Corresponding author: Nurfadhlina Mohd Sharef (nurfadhlina@upm.edu.my)

This work was supported by the Intelligent Computing Research Group, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia, under the Airforce Office of Scientific Research (AFOSR) on multi-objective deep recurrent reinforcement learning research under Grant FA2386-18-1-4079.

ABSTRACT Question-answering chatbots have tremendous potential to complement humans in various fields. They are implemented using either rule-based or machine learning-based systems. Unlike the former, machine learning-based chatbots are more scalable. Sequence-to-sequence (Seq2Seq) learning is one of the most popular approaches in machine learning-based chatbots and has shown remarkable progress since its introduction in 2014. However, chatbots based on Seq2Seq learning show a weakness in that it tends to generate answers that can be generic and inconsistent with the questions, thereby becoming meaningless and, therefore, may lower the chatbot adoption rate. This weakness can be attributed to three issues: question encoder overfit, answer generation overfit, and language model influence. Several recent methods utilize multitask learning (MTL) to address this weakness. However, the existing MTL models show very little improvement over single-task learning, wherein they still generate generic and inconsistent answers. This paper presents a novel approach to MTL for the Seq2Seq learning model called SEQ2SEQ++, which comprises a multifunctional encoder, an answer decoder, an answer encoder, and a ternary classifier. Additionally, SEQ2SEQ++ utilizes a dynamic tasks loss weight mechanism for MTL loss calculation and a novel attention mechanism called the comprehensive attention mechanism. Experiments on NarrativeQA and SQuAD datasets were conducted to gauge the performance of the proposed model in comparison with two recently proposed models. The experimental results show that SEQ2SEQ++ yields noteworthy improvements over the two models on bilingual evaluation understudy, word error rate, and Distinct-2 metrics.

INDEX TERMS Sequence to sequence learning, natural answer generation, multitask learning, attention mechanism.

I. INTRODUCTION

Chatbots are online computer systems that can mimic humans and converse with humans using natural language [1], [2]. Chatbots can be broadly categorized into two types, namely transactional and conversational [3], [4]. The former ones help humans perform specific tasks to achieve a specific goal, such as booking a hotel, flight, or even performing a financial transaction [5]. The latter ones act as companions for humans and can be further classified as either casual chatbots or question-answering chatbots [3], [6]–[9].

Casual chatbots are based on language models and can interact with users through a series of natural language dialogs and interactions, either verbally or textually. On the other hand, question-answering chatbots aim at automatically providing answers to a single or series of questions posed by users in natural language in a specific domain or area such as customer support [12]. Such a chatbot can be modeled by means of three approaches: answer extraction, answer generation, and answer selection (Figure 1). Answer extraction

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Gaggero.

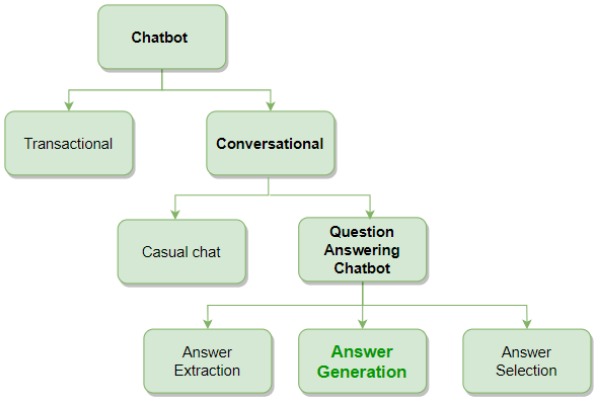

FIGURE 1. Chatbot types.

refers to the process of extracting the answer from a text for a question. Answer selection refers to the selection of the best answer for a question from a list of answers. Answer generation refers to the task of generating the answer to a question. The generated answer can be a new answer, which is not seen in the existing training dataset.

There are two popular approaches in conversational chatbot modeling namely Transformer network-based models such as [13]–[15] and recurrent neural network (RNN)-based sequence to sequence learning (Seq2Seq) models such as [16]–[22]. The Transformer network is based on the feed-forward network [11], wherein sentences are processed as a whole rather than word by word by utilizing a self-attention mechanism, which can be highly parallelized. However, Transformer-based models need to handle sequence processing with positional embeddings to encode information related to a specific position, and this requires very high computational and memory cost because of their quadratic memory usage and computational time complexity $O(N^2)$ costs where N is the sequence length [10], [14], [23]–[27].

The other popular and more suitable method for sequence processing such as question answering is the RNN-based Seq2Seq method [1], [28]. RNNs view input as a chain structure whereby the next output depends on the previous hidden state (sequential by design), thus making RNN-based models have the capability to process sequences of variable length. RNN-based models cannot be parallelized and thus can be slow while processing long sequences [11]. **However, unlike the Transformer network, RNN-based models require only linear number of $O(N)$ of operations where N is the sequence length and thus do not require high computational and memory cost. This shows RNN-based Seq2Seq method is still worthwhile to be investigated and has great potential. It is to be noted that the Seq2Seq learning approach has been extensively researched for both single-turn and multiturn conversations [1], [16]–[22], [28].**

We focused on a single-turn (non-hierarchical) question-answering (answer generation) using the RNN-based

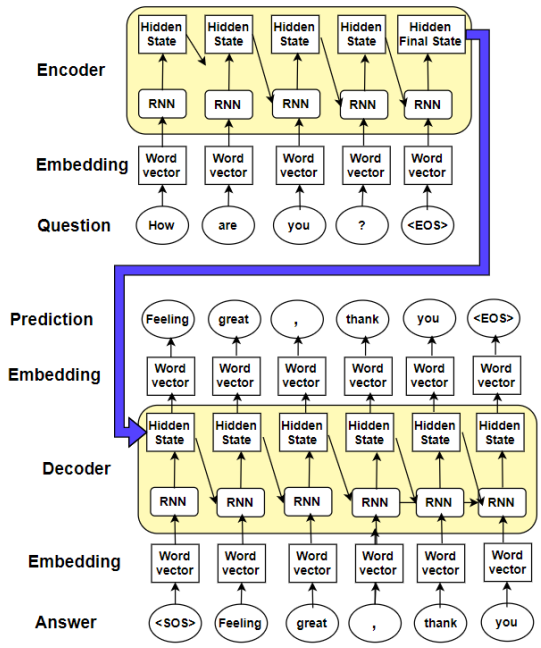

FIGURE 2. Illustration of Seq2Seq learning. The question is converted to embedding and then encoded to hidden states. Encoder hidden states are used by the decoder to generate the answer.

Seq2Seq learning under Multitask Learning (MTL) framework, similar to what is defined in our key benchmark paper [18]. MTL is a machine learning approach where multiple tasks are learned to improve answer generation quality. MTL has shown success in many applications of machine learning including NLP [29], [30]. MTL in the Seq2Seq model was first explored for machine translation problems [29], [31] and the success lent an impetus to other researchers [18], [20], [32]–[36] to explore it for chatbot answer generation to address the issue of generic and inconsistent answers by Seq2Seq model. Seq2Seq learning under MTL has considerable potential because the MTL framework provides an efficient mechanism to integrate multiple enhancements as discussed further in this paper. [18] is particularly of interest to us because it is the only paper known to us that utilized the MTL framework for answer generation without the requirement of a secondary dataset. All other MTL frameworks such as [20], [32]–[36] require an additional dataset, which may not available in all scenarios.

The Seq2Seq method utilizes an encoder and decoder architecture [37], [38]. The encoder, which consists of an RNN, aims to represent the meaning of the question sentence by encoding the question sentence into a dense vector called hidden states. Subsequently, the decoder, which is another neural network, aims to generate an answer sentence based on the encoder's hidden states. Figure 2 shows a typical Seq2Seq model.

However, findings of a number of studies [18]–[22] have demonstrated that the Seq2Seq method [37] tends to generate frequently occurring words in the answer, thereby

compromising the quality of the generated answer. Generated answers may be meaningless or irrelevant to the question and, as a result, conversations with chatbots can be meaningless, abruptly terminated by users, and eventually lower the chatbot adoption rate. This weakness can be attributed to three (3) key issues: language model influence, answer generation overfit, and question encoder overfit. These issues are described in further detail in the following sub-sections.

A. LANGUAGE MODEL INFLUENCE

The decoder in a Seq2Seq model, typically an RNN, also behaves like a language model. A language model refers to the ability to generate the next word on the basis of previously generated words or words. **As the decoding progresses, the language model influence of the decoder becomes stronger than the influence of the question.** Consequently, the decoder may generate answers that are irrelevant to the question.

B. ANSWER GENERATION OVERFIT

Seq2Seq method learning occurs by optimizing the cross-entropy loss function to find the best sequence of words that form the answer. The goal of training is to minimize the loss during training. However, **uneven word frequency in data causes the model to generate frequently occurring words to minimize the loss during training, which results in an answer generation overfit**. The answer-generation overfit causes the model to produce frequently occurring words learned from the dataset.

C. QUESTION ENCODER OVERFIT

Question-answering models are typically trained with specific datasets in a domain, such as customer support logs or question–answer pairs of an academic subject. However, **the availability of these data in these specific domains may be limited. The question encoder overfit refers to the situation in which the question encoder becomes overfit owing to limited training data. Overfit occurs because the model generates question encodings to minimize loss during training. Question encoder overfit causes the model to suffer when handling unseen questions**.

In addition, most of the existing work found in the literature focuses only on addressing either the overfitting issue or language model influence issue, but not both, thereby leaving a gap and a great opportunity to address the weakness in a more holistic manner.

In this paper, we study how to address all three issues in an MTL setting. We introduce the SEQ2SEQ++, MTL-based Seq2Seq model using RNN, which consists of a multifunctional encoder (MFE), an answer decoder (AD), an answer encoder (AE), and a ternary classifier (TC). MFE performs question encoding, first-word prediction, and last-word prediction. AD performs answer generation, and AE performs answer decoding. The TC performs a three-class classification of answers for a given question. Additionally, our method utilizes a dynamic task loss weight scheme for MTL loss calculation and a novel attention mechanism called the comprehensive attention mechanism (CAM) for answer generation.

TABLE 1. Explanations on notation for Figure 3.

Notation	Meaning
f_{GRU}	Gated Recurrent Unit (GRU) (Cho et. al. , 2014) function. GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size dense vectors)
$Embed_{t-1}$	Previously generated word embedding (at timestep t-1)
$h_1,h_2,...,h_T$	Hidden states of the question encoder
s_t	Hidden states of the decoder at timestep t
f_{ATT}	Attention weight computation function
f_c	Context vector computation function
ATT_t	Attention weights at timestep t
C_t	Question context vector
O_t	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t

The following works were considered for performance benchmarking where two datasets namely NarrativeQA [39] and SQuAD [40] are utilized:-

i) MTL-BC: An MTL model that utilizes a binary classifier as the auxiliary task and fixed task loss weighting, as proposed in [18]. This is our key benchmark paper.
ii) STL: Single-task baseline Seq2Seq learning with global attention mechanism [41]. This single-task learning (STL) model is used as control method.
iii) MTL-LTS: A sequential MTL model that utilizes a separate network to predict the first word, as proposed by [42]. This is our key benchmark paper. This work aims to compare effects of parallel MTL training proposed in our work.

The key contributions of this paper are:-

i) A new MTL-based Seq2Seq model, called SEQ2SEQ++, for question-answering
ii) Dynamic task loss weight mechanism for MTL: A new computation method to calculate the task loss weights automatically and dynamically
iii) CAM: a novel attention mechanism for answer generation

II. RELATED WORK

In this section, we review the methods found in the extant literature and identify the gaps.

A. LANGUAGE MODEL INFLUENCE ISSUE

A typical approach to address the language model influence of the decoder is the attention mechanism. An attention mechanism is a method that allows the decoder to focus on certain parts of a question to decode the answer to that question. The most prevalent and extensively utilized attention mechanism is the global attention mechanism (Figure 3, Table 1), which computes the attention weights in accordance with the encoder's hidden states and decoders' last hidden

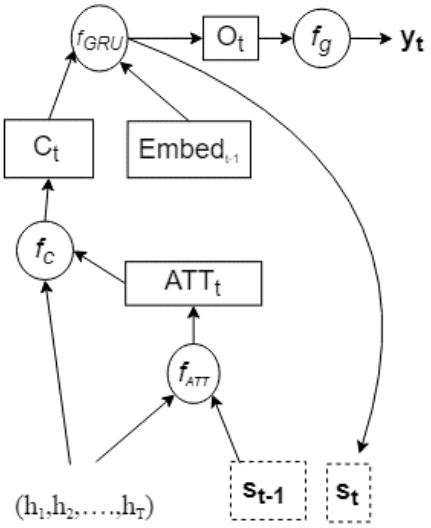

FIGURE 3. Answer generation based on the global attention mechanism which uses only on the final hidden state of the decoder (st-1).

state [41]. These computed attention weights (ATT_t) are then used to compute the context vector C_t), which is then used for subsequent computation to eventually generate the answer.

Several other attention mechanisms have been proposed, such as the local attention mechanism [43], hybrid attention mechanism [44], and weighted attention mechanism [45]. However, similar to the global attention mechanism [41], these attention mechanisms [43]–[45] also focus only on the encoder's hidden states and the decoder's final hidden state at each decoding step. Although the decoder's final hidden state represents all hidden states of previous time steps, the representation of the answer words generated in earlier time steps becomes diluted as it progresses in time and thus increases the decoder's language model influence, which tends to generate irrelevant answers. This leaves a gap in identifying an attention mechanism that can consider all hidden states of the decoder during decoding to address the language model influence more effectively.

B. ANSWER GENERATION OVERFIT ISSUE

The answer-generation overfit can be addressed by adding one or more regularization terms to the cross-entropy loss function to compute a new loss to be backpropagated.

One existing approach is to train the Seq2Seq method in a reinforcement learning or adversarial framework. However, reinforcement or adversarial learning requires custom reward functions or human interactions [46], which render this approach less practicable for application in cross-domain problems.

Another more practical approach to address overfitting is training the Seq2Seq method in a "multi-task learning," i.e., MTL framework. MTL is a machine learning approach in which multiple tasks are learned in parallel to improve the main task. For question-answering, the main task to be improved is the answer-generation task. In MTL-based Seq2Seq learning, in addition to the answer generation task (main task), other tasks (usually referred to as auxiliary tasks) are introduced during model training. The losses from the auxiliary tasks are used as a regularization term to reduce the overfitting of the main task of answer generation. Equation (1) shows a generic form of multitask loss calculation where L_{MTL} is the total model loss, L_{ag} is the answer-generation task loss, L_n refers to the loss of an n-th auxiliary task, and α refers to the task loss weight for each task in MTL. The task loss weights determine the extent to which the task influences learning and the extent to which the task will be learned. Therefore, it is vital to identify suitable auxiliary tasks. Moreover, in all existing MTL works for Seq2Seq learning, a fixed-weight scheme is used to calculate the MTL loss. In this fixed weight scheme, the auxiliary task is typically assigned a small value, such as 0.001, 0.01, or 0.1 [18]. However, determining the weight for the auxiliary task loss is not an easy task, given that there are no specific rules or formulas to determine the actual value to be used. Arbitrary values must be assigned and tested before the final weight for a task can be identified. Researchers have to perform numerous experiments on the task loss weights on a trial-and-error basis before settling for a specific value. In addition, training different datasets may require different values to be used. The task loss weight for a dataset may not be effective for another dataset. This trial-and-error approach is time-consuming and inefficient. It becomes even worse or nearly impossible if there are more than two tasks. This leaves a gap in identifying a more efficient and effective approach to determine the auxiliary task loss weights for an MTL.

$$L_{MTL} = \alpha_{ag} L_{ag} + \alpha_n L_n \tag{1}$$

C. QUESTION ENCODER OVERFIT ISSUE

There are three approaches for addressing the *question encoder overfit* issue. The first approach is to provide additional embeddings or additional encodings of supporting information such as emotions, topic, or facts that accompany the question [19]–[22], [47], [48]. The underlying idea for this approach is to reduce the overfitting of the encoder so that it can generate a richer representation of the question to be passed to the decoder to furnish relevant answers. However, these approaches are skewed to a specific goal and are also dependent on additional inputs such as facts, topics, or emotions, which may not be available for all question-answering scenarios or datasets. This leaves a gap in identifying a method that can reduce the question encoder overfitting without depending on any additional input.

The second approach is the MTL approach, wherein a Seq2Seq model is trained to perform answer generation and another task such as answer classification [18], [20]. The underlying idea for this approach is to share the encoding of the question encoder to perform both tasks so that the question encoder overfit can be reduced. For example, [6] utilized binary question–answer classification as an

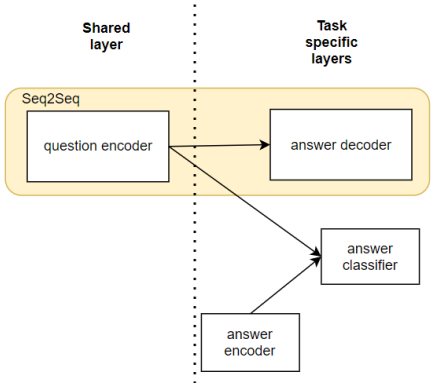

FIGURE 4. Multitask Learning consisting of answer generation and answer classification.

TABLE 2. Sample qestion and answer.

Question	What did you have for breakfast ?
Answer (exact answer/fully correct answer	I had half-boiled egg and toasted bread for breakfast.
Partially correct answer -#1	I had half-boiled egg for breakfast
Partially correct answer - #2	I had toasted bread for breakfast

auxiliary task (Figure 4). Binary classification refers to a classification in which an answer is classified as either correct or incorrect. However, classifying an answer as right or wrong only (binary classification) may not be a natural classification method. In NLP, a generated answer can also be categorized as partially correct.

Referring to Table 2 as an example, the question is ''What did you have for breakfast?'' and the exact answer is ''I had half-boiled egg and toasted bread for breakfast.'' This is a fully correct answer. Suppose ''I had toasted bread for breakfast'' is given as an answer. It is not fully correct, but at the same time, it is not completely wrong. In this scenario, the given answer can be considered a partially correct answer. However, in binary classification, this answer is classified as incorrect. In line with this argument, binary classification is not a ''natural'' classification of answers, which leaves a gap in identifying an auxiliary task with a more natural answer classification to be used in MTL-based answer-generation model training.

The third approach to address the question encoder overfit is to train the Seq2Seq model using a two-phased (sequential) training approach [42]. During phase 1, this model learns to perform first-word prediction, whereas during phase 2, the model is trained to predict the answers, except for the first word. The idea of first-word prediction as an auxiliary task in MTL is a good idea to reduce model overfit. However, performing sequential MTL suffers from an issue referred to as a negative transfer. It is a situation where learning of the first task may negatively affect the learning of the second task.

TABLE 3. Gaps in existing works.

Issue	Gaps
Language model influence	Attention computation doesn't consider all previously generated hidden states during decoding
Answer generation overfit	MTL loss calculation uses a fixed task loss weights for all tasks which is not efficient nor effective
Question encoder overfit	Binary classification as auxiliary task is not a natural classification. Sequential learning based MTL for first-word prediction suffers from negative transfer issue.

This leaves a gap in identifying a more suitable approach to add the first-word prediction as a task in the MTL framework.

Table 3 summarizes the gaps identified in the related works, as discussed.

III. PROPOSED METHODS AND MODEL

In this section, we describe our proposed model, SEQ2SEQ++ (Figure 5, Table 4), to address the issues and gaps discussed in the previous section. SEQ2SEQ++ integrates four newly proposed methods: the CAM, DL weighting scheme, TC, and MFE in a single model.

The four newly proposed methods in this work are CAM, DL, TC, and MFE.

i) CAM is an attention computation method proposed to address the language model influence issue.
ii) DL is a dynamic tasks loss weight computation method implemented during SEQ2SEQ++ training and proposed to address the answer-generation overfit issue.
iii) MFE and TC are proposed to address the question encoder overfit issue.

By integrating all these methods, SEQ2SEQ++ performs four tasks in parallel. These tasks are answer generation, ternary classification, first-word prediction, and last-word prediction tasks.

The details of each method are discussed in the following sections.

A. COMPREHENSIVE ATTENTION MECHANISM

In this work, a new attention mechanism called the ''comprehensive attention mechanism'' or CAM, which considers all the decoder's previous hidden states during attention weight computation, is proposed to address the language model influence more effectively.

In CAM, the attention weights are computed in accordance with all the encoder's hidden states and the sum of all the previous hidden states of the decoder. These computed attention weights are then used to compute the context vector. Subsequently, the decoder utilizes the context vector to generate an answer. The CAM-based decoding steps are the same as a typical decoding process to generate the answer, except for the computation of the attention weights.

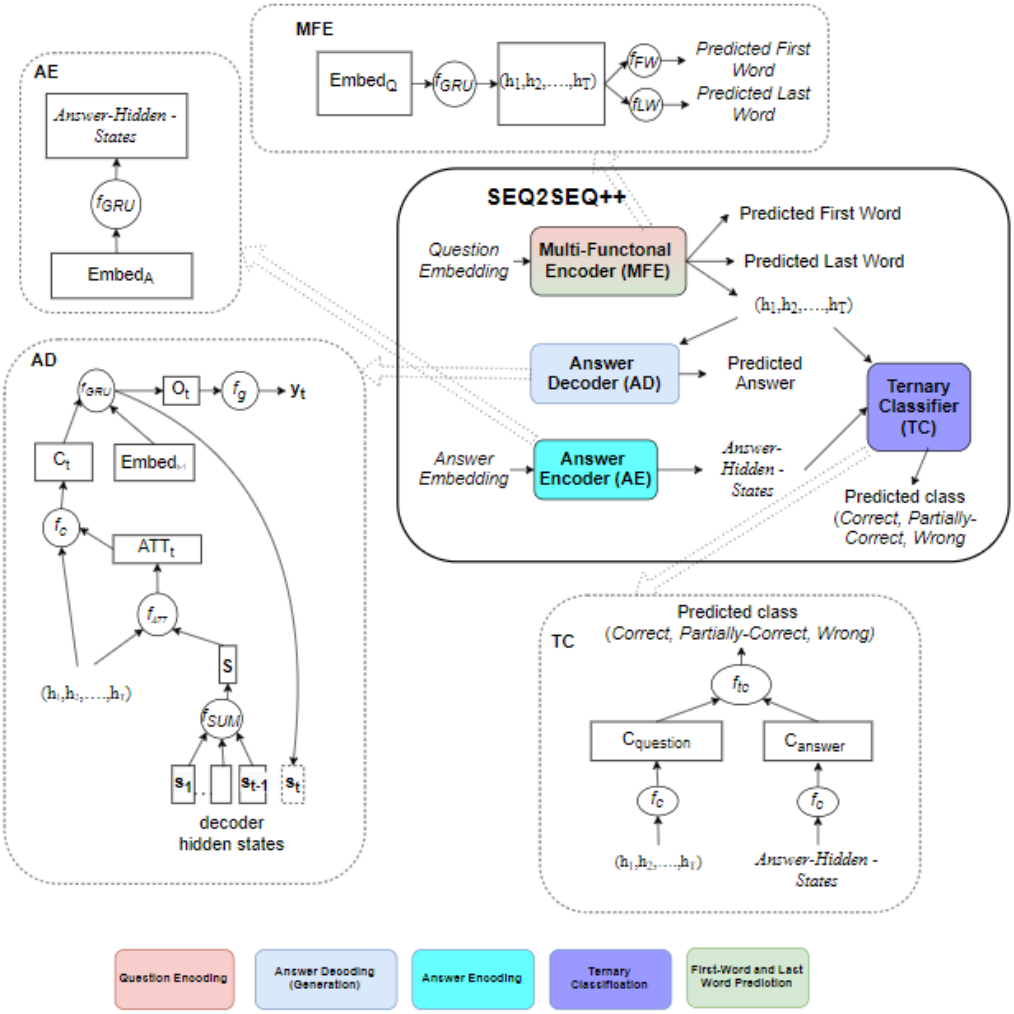

FIGURE 5. SEQ2SEQ++ model. MFE performs question encoding and first word and last word predictions. AE perform answer encoding. TC performs ternary classification of the answer. AD performs answer decoding (answer generation) based on $S = \sum (s1, s2, \ldots, st - 1)$ (Comprehensive Attention Mechanism).

TABLE 4. Explanations on notation for the SEQ2SEQ++ model.

Notation	Meaning
f_{GRU}	Gated recurrent unit (GRU) (Cho et. al., 2014) function. GRU function is a non-linear transformation that transforms embedding into hidden states (fixed-size dense vectors)
f_{FW}	First-word prediction function
f_{LW}	Last-word prediction function
$Embed_x$	Embedding of either question or answer
$h_1,h_2,...,h_T$	Hidden states of the question encoder
f_{SUM}	Matrix summation operation
$s_1,s_2,...,s_T$	Hidden states of the decoder
f_{ATT}	Attention weight computation function
f_c	Context vector computation function
f_{TC}	Ternary classification function
C_t	Question context vector
O_t	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t

The decoding steps are as follows:-

i) First, during decoding, at time t, the decoder scores the alignment based on all the question encoder hidden states (h_1 to h_T) and the summation of all the decoder's previous hidden states (S) (Eq. 2)

$$ATT_t = Softmax(Relu((W_Q) \cdot (h_1, h_2, \ldots, h_T) + (W_{t1}) \cdot (S))) \quad (2)$$

where:
$h_1, h_2, \ldots, h_T$ are the hidden states of the encoder
t = decoding timestep
$S = \sum (s_1, s_2, \ldots, s_{t-1})$

ii) Second, the context vector for decoding at time step $t(C_t)$ is generated (Eq. 3)

$$C_t = \sum (ATT_t * (h_1, h_2, \ldots, h_T)) \quad (3)$$

where C_t is a weighted average context vector for answer prediction.

iii) Next, the output context vector (O_t) and decoder's hidden state at time step t, (s_t) are computed based on the concatenation of the generated context vector (C_t) and supplied ground truth (y_{t-1}) using GRU transformation (f_{GRU}). O_t is used for the prediction of the next word and s_t is utilized for subsequent decoding steps (Eq. 4).

$$O_t, s_t = f_{GRU}(concat(C_t, Embed_{t-1}) \quad (4)$$

where
O_t is the output context vector generated by the GRU transformation
s_t is the hidden state at time step, t generated by the GRU transformation
$Embed_{t-1}$ is the embedding of the supplied ground truth word or the previous predicted word (y_{t-1}).

iv) Finally, the conditional probability of the next token is computed in accordance with the output vector $p(y_t)$ (Eq. 5).

$$p(y_t) = (W_{t2}) \cdot (O_t) \quad (5)$$

where W_x are matrixes (learnable parameters) that is be trained in the model.

B. MULTIFUNCTIONAL ENCODER

MTL provides the means to add auxiliary tasks that can be trained simultaneously with answer-generation tasks. By utilizing auxiliary tasks that share the question encoder, it forces the question encoder to balance and fine-tune the encodings for all receiving tasks. This step ensures that all receiving tasks can mitigate their respective prediction losses, such as answer generation loss or answer classification loss. Because each task in an MTL contributes to the overall model loss, mitigating each task's loss to is very important. Thus, by sharing the question encoder, the question encoder overfit will eventually be reduced. This can ensure that the answer generation bias toward high-frequency words is reduced, and thus a more meaningful answer can be generated.

The MFE is proposed to take advantage of the parallel MTL. The question encoder is shared with two additional tasks: first- and last-word prediction. MFE performs its tasks in two stages.

i) First, it takes in question embedding and generates question encoding using the GRU transformation [37] denoted as f_{GRU} in Figure 5 and Eq. 6.
ii) Next, additional computations on the question encoding (hidden states) are performed to make the first word (shown as f_{FW} in Figure 5), as shown in (Eq. 7) to (Eq. 9) and last-word predictions (shown as f_{LW} in Figure 5), respectively, as shown in (Eq. 10) to (Eq. 12).

The following computations are performed during the first-word prediction (f_{FW}):-

i) First, self-attention weights are computed (Eq. 7).
ii) Next, the question context vector for first-word prediction (C_{FW}) is computed as the weighted average of the question encoder hidden states (Eq. 8).

Finally, the probabilities of each word in the vocabulary to be the first word are computed (Eq. 9).

Similar computations are performed to predict the final word. This is denoted as the f_{LW} function in Figure 5. The computations are shown in Eq. 10–12.

$$(h_1, h_2, \ldots, h_T) = f_{GRU}\left(Embed_Q\right) \quad (6)$$

$$ATT_{FW} = Softmax\left(Relu\left((W_{FW1}) \cdot (h_1, h_2, \ldots, h_T)\right)\right) \quad (7)$$

$$C_{FW} = \sum (ATT_{FW} * (h_1, h_2, \ldots, h_T) \quad (8)$$

$$p(first - word) = Softmax\left(W_{FW2}\right) \cdot (C_{FW}) \quad (9)$$

$$ATT_{LW} = Softmax\left(Relu\left((W_{LW1}) \cdot (h_1, h_2, \ldots, h_T)\right)\right) \quad (10)$$

$$C_{LW} = \sum (ATT_{FW} * (h_1, h_2, \ldots, h_T) \quad (11)$$

$$p\left(l_{ast} - word\right) = Softmax\left(W_{LW2}\right) \cdot (C_{LW}) \quad (12)$$

where:
h are the hidden states for all time steps,
T is the question sequence length,
$Embed_Q$ is the question embedding
$h_1, h_2, \ldots, h_T$ are the hidden states of the encoder

t = decoding timestep

ATT_{FW} and ATT_{LW} are the self-attention weights

C_{FW} is a weighted average context vector for first word prediction

W_x are matrixes (learnable parameters) that is be trained in the model

W_{FW1} and W_{LW1} are learnable matrix

C_{LW} is weighted average

C. TERNARY CLASSIFIER

Similar to MFE, a TC is proposed to take advantage of parallel MTL to reduce the question encoder overfit. The TC classifies a question–answer pair as ''correct,'' ''partially correct,'' or ''wrong.''

As illustrated in Figure 5, the TC performs the question–answer classification task based on the question encodings that are generated by the MFE, and answer encoding, which is generated by the answer encoder (AE). Eq. 13 shows the answer generation where $Embed_A$ can be embedding for correct, partially- correct or wrong answers. *Answer-Hidden-States* can be one of the below: -

i) $a_1, a_2 \ldots, a_U$, hidden states for correct answer, U is the correct answer length

ii) $b_1, b_2 \ldots, b_V$, hidden states for partially correct answer, V is the partially correct answer length

iii) $c_1, c_2 \ldots, c_W$, hidden states for wrong answer, W is the wrong answer length.

The question and answer context vectors are computed as the weighted average of the question and answer encodings, respectively (Eq. 14) to (Eq. 21). The probability for each classification class is then computed on the concatenated question and answer vectors (Eq. 22) to (Eq. 24).

$$\begin{aligned}
&Answer\text{-}Hidden\text{-}States \\
&\quad = f_{GRU}\left(Embed_A\right) && (13)\\
AttQ &= Softmax\left(Relu\left(W_Q\right)\cdot\left(h_1, h_2, \ldots, h_T\right)\right) && (14)\\
C_Q &= \sum\left(Att_Q * \left(h_1, h_2, \ldots, h_T\right)\right) && (15)\\
Att_C &= Softmax\left(Relu\left(W_{Cl}\right)\cdot\left(a_1, a_2, \ldots, a_U\right)\right) && (16)\\
C_C &= \sum\left(Att_C * \left(a_1, a_2, \ldots, a_u\right)\right) && (17)\\
Att_P &= Softmax\left(Relu\left(W_{PI}\right)\cdot\left(b_1, b_2, \ldots, b_v\right)\right) && (18)\\
C_P &= \sum\left(Att_P * \left(b_1, b_2, \ldots, b_v\right)\right) && (19)\\
Att_W &= Softmax\left(Relu\left(W_{WI}\right)\cdot\left(c_1, c_2 \ldots, c_W\right)\right) && (20)\\
C_W &= \sum\left(Att_W * \left(c_1, c_2 \ldots, c_W\right)\right) && (21)\\
p(class &= \text{'}Correct\text{'}) \\
&= Softmax\left(\left(W_{C2}\right)\cdot\left(concat\left(C_Q, C_C\right)\right)\right) && (22)\\
p(class &= \text{'}Partially-Correct\text{'}) \\
&= Softmax\left(\left(W_{P2}\right)\cdot\left(concat\left(C_Q, C_P\right)\right)\right) && (23)\\
p(class &= \text{'}Wrong\text{'}) \\
&= Softmax\left(\left(W_{W2}\right)\cdot\left(concat\left(C_Q, C_W\right)\right)\right) && (24)
\end{aligned}$$

where:

ATT_Q, ATT_C and ATT_P are the self-attention weights

C_Q is the computed question context vector at time step, t

$a_1, a_2, \ldots, a_U$ are the hidden states of the decoder for correct answers

U is the correct answer length

C_C is the computed correct answer context vector at time step, t

$c_1, c_2 \ldots, c_W$, hidden states for wrong answer

W is the wrong answer length

$b_1, b_2, \ldots, b_V$ are the hidden states of the decoder for partially correct answers

V is the partially correct answer length

C_P is the computed partially correct answer context vector at time step, t

C_W is the computed wrong answer context vector at time step, t.

D. DYNAMIC TASKS LOSS WEIGHT SCHEME

In this work, a task loss weight scheme called the ''dynamic tasks loss weights scheme'' is proposed. In this scheme, during model training, the task loss weights for each task (*answer generation, answer classification, first-word prediction, and last-word prediction*) are automatically recalculated for the second epoch. The new weights are based on the relative loss of each task in comparison with the total loss of all tasks during each epoch (Eq. 25), where α_n is the task loss weight, L_{Tn} is the loss of task-n, and N is the total number of tasks. The weights represent the percentage contribution of each task to the overall MTL loss. Therefore, the sum of all weights should be 1, which represents 100%.

$$\boldsymbol{\alpha_n} = \frac{\boldsymbol{L_{Tn}}}{\sum_{n=1}^{N} \boldsymbol{L_{Tn}}} \tag{25}$$

Algorithm 1 delineates the steps to perform the SEQ2SEQ++ model training, which utilizes the DL scheme. First, the shared question encoder performs question encoding, and subsequently the first- and last-word predictions. The model then computes the loss for each prediction (L_{fw} and L_{lw}). The question encoding is then passed to the answer decoder to generate an answer using CAM. This model then computes the answer-generation loss (lag). In addition, the answer encoder performs the answer encoding. Both the question encoding and answer encoding are then passed to the TC to perform question–answer classification and subsequently compute the ternary classification loss (L_{tc}). The model then computes the MTL loss (LMTL) and updates its weights (i.e., parameters). The formula for the MTL loss calculation is shown in step 1.7 in Algorithm 1. Finally, the model computes the new task loss weights to be used for the next epoch. These steps are performed for each batch of the question–answer pairs for the total epoch count, as defined for training.

The variables α_{ag}, α_{tc}, α_{fw}, and αl_w represent the task loss weight for answer generation, the task loss weight ternary classification, the task loss weight for the first-word prediction, and the task loss weight for last-word prediction tasks. At the start of the training, each weight is initialized to 0.25.

Algorithm 1 SEQ2SEQ++ Training Algorithm

Input: {Question (X), Answer (Y), Label (L), First Word (FW), LastWord (LW)} quintuplets, Maximum answer length (T), Maximum Epoch (E)

Steps:

Initialize Multi-functional Encoder (MFE), Answer Encoder (AE), Answer Decoder (AD) and Ternary Classifier (TC)

Initialize each losses $L_{MTL} = L_{ag} = L_{tc} = L_{fw} = L_{lw} = 0$

Initialize each task loss weights $\alpha_{ag} = \alpha_{tc} = \alpha_{fw} = \alpha_{lw} = 0.25$

For epoch 1 to Number of Epochs

1. For batch 1 to Number of Batches Do
   - 1.1 Perform question encoding
   - 1.2 Predict First Word and compute the first-word prediction losses (L_{fw})
   - 1.3 Predict Last Word and compute the last-word prediction losses (L_{fw})
   - 1.4 Perform answer encoding
   - 1.5 Perform ternary classification and compute ternary classification loss (L_{tc})
   - 1.6 Perform answer generation using CAM and compute answer generation loss (L_{ag})
   - 1.7 Compute multi-task loss:

$$L_{MTL} = \alpha_{ag} * L_{ag} + \alpha_{tc} * L_{tc} + \alpha_{fw} * L_{fw} + \alpha_{lw} * L_{lw}$$

   - 1.8 Update the model parameters (neural network weights)
   - 1.9 For each task, $tk \subseteq \{ag, tc, fw, lw\}$: Calculate the average loss ($L_{tk\text{-}avg}$)

End for Batch Looping

2. Calculate the total average loss for all tasks: $L_{avg\text{-}total} = L_{ag\text{-}avg} + L_{tc\text{-}avg} + L_{fw\text{-}avg} + L_{lw\text{-}avg}$
3. For each task, $tk \subseteq \{ag, tc, fw, lw\}$
   - 3.1 Calculate new task loss weight: $\alpha_{tk} = L_{tk\text{-}avg}/(L_{avg\text{-}total})$

End for Epoch Looping

Output: Trained SEQ2SEQ++ model

During each epoch, all the task loss weights (α_{ag}, α_{tc}, α_{fw}, and α_{lw}) are recalculated and updated for use in the next epoch (step 3 of Algorithm 1).

Table 5 presents a sample calculation. The total MTL loss is 4, and the answer generation loss is 2. Therefore, the new task loss weight for answer generation (α_{ag}) is $2/4 = 0.5$. The same calculation is performed to calculate the new task loss weights for all tasks. The new task loss weights for each task will be proportional to the total model loss. This means that the influence of each task in each epoch is dynamically determined by the task loss in the previous epoch. Thus, model overfitting can be effectively reduced using this dynamic proportional task loss weight during training as compared to using a fixed task loss weight approach, as in typical MTL learning.

TABLE 5. SEQ2SEQ++ - Sample task loss weights calculation.

Tasks Losses	
Answer generation loss ($L_{ag\text{-}avg}$)	2
Ternary classification loss ($L_{tc\text{-}avg}$)	1
First word prediction loss ($L_{fw\text{-}avg}$)	0.5
Last word prediction loss ($L_{lw\text{-}avg}$)	0.5
Calculations	
Total loss ($L_{avg\text{-}total}$)	4
Answer generation task loss weight (α_{ag})	2/4 = **0.5**
Ternary classification task loss weight (α_{tc})	¼ = **0.25**
First-word prediction task loss weight (α_{fw})	0.5/4 = **0.125**
Last word prediction task loss weight (α_{lw})	0.5/4 = **0.125**

TABLE 6. Experimented models.

Models Involved	Purpose of Model
MTL-BC [18]	Key benchmark model. This is a parallel model, which performs answer generation and answer classification.
STL [41]	Second benchmark model. This is a single-task model, which performs only answer generation.
MTL-LTS [42]	Third benchmark model. It is a sequential MTL model, which performs answer generation and first-word prediction.
MTL-BC-CAM	To study the effectiveness of CAM to address language model influence issue.
MTL-BC-DL	To study the effectiveness of DL to address the answer generation overfit issue.
MTL-TC	To study the effectiveness of ternary classification to address the question encoding overfit issue.
MTL-MFE	To study the effectiveness of MFE to address the question encoding overfit issue.
SEQ2SEQ++	To study the effectiveness of integrating all our proposed methods (CAM, DL, MFE, and TC) in a single model.

IV. EXPERIMENT AND DISCUSSION

We developed eight models, and trained and tested each of them on two datasets NarrativeQA [39] and SQuAD [40] to gauge the effectiveness of our proposed methods to generate meaningful and relevant answers for each dataset. All except the STL model are MTL-based models. The STL model is a single-task method, which has the answer-generation task only and is used as a control method. The results of the experiments are described in detail in this section.

A. MODELS

The models (summarized in Table 6) utilized for these experiments are:-

i) MTL-BC: The MTL-BC model is an MTL-based Seq2Seq model that utilizes the global attention mechanism [41] during decoding and a binary classifier as the auxiliary task [18]. This model is trained using

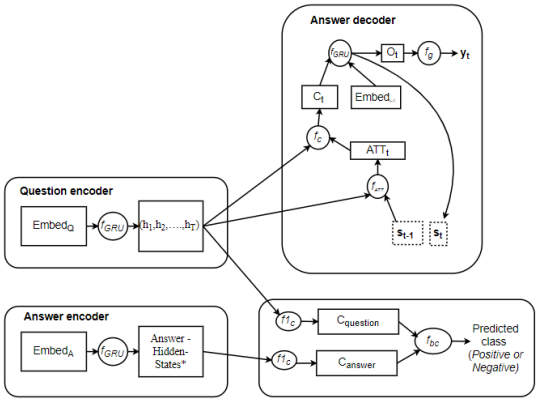

FIGURE 6. MTL-BC model.

TABLE 7. Explanations on notation for the MTL-BC model.

Notation	Meaning
f_{GRU}	GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size densed vectors) [37]
$Embed_{t-1}$	Previously generated word embedding (at timestep t−1)
$Embed_Q$	Question embedding
$Embed_A$	Answer embedding
$h_1,h_2,...,h_T$	Hidden states of the question encoder
s_t	Hidden states of the decoder at timestep t
f_{ATT}	Attention weight computation function
f_c,fl_c	Context vector computation function
ATT_t	Attention weights at timestep t
C_t	Question context vector
O_t	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t
f_{bc}	Binary classification function

Algorithm 2. It utilizes a fixed task loss weight scheme for multitask loss calculation during training, whereby the value of 0.1 is assigned as the binary classification task loss weight. The architecture of the MTL-BC model is shown in Figure 6. The notations are summarized in Table 7. This model is the second and key benchmark model.

ii) STL: The STL model is a single-task Seq2Seq learning model as proposed by Bahdanau *et al.* [41] and utilizes the global attention mechanism [41] and is trained using Algorithm 2. This model is the first benchmark model. The architecture of the STL model is shown in Figure 7. The notations are summarized in Table 8.

iii) MTL-LTS: The MTL-LTS model is the third benchmark for this research work. It is a sequential MTL model with a global attention mechanism that utilizes a separate network as the auxiliary task to predict the first word, as proposed in [42]. This model is trained using Algorithm 4. The architecture of the MTL-LTS model is shown in Figure 8. The notations are summarized in Table 9.

Algorithm 2 MTL-BC Model Training

Input: {Question (X), Answer(Y), Label(L))} triplets, Maximum answer length (T), Maximum Epoch (E)

Steps:

Initialize Question Encoder, Answer Encoder (AE), Answer Decoder (AD) and Binary Classifier (BC)

Initialize each loss $L_{MTL} = L_{ag} = L_{bc} = 0$

Initialize each task loss weights $\alpha_{ag} = 1.0$; $\alpha_{bc} = 0.1$

For epoch 1 to Number of Epochs

For batch 1 to Number of Batches Do

1. Perform question encoding: Generate the hidden states for each timestep
2. Perform answer encoding: Generate the hidden states for each timestep
3. Perform binary classification and compute classification loss (L_{bc})
4. Perform answer generation using the weighted hidden states from Encoder and *its own previously generated hidden state*, and Compute answer generation loss (L_{ag})
5. Compute multi-task loss:
   i) $L_{MTL} = \alpha_{ag} * L_{ag} + \alpha_{bc} * L_{bc}$
6. Update the model parameters (neural network weights)

End for Batch Looping

End for Epoch Looping

Output: Trained MTL-BC model

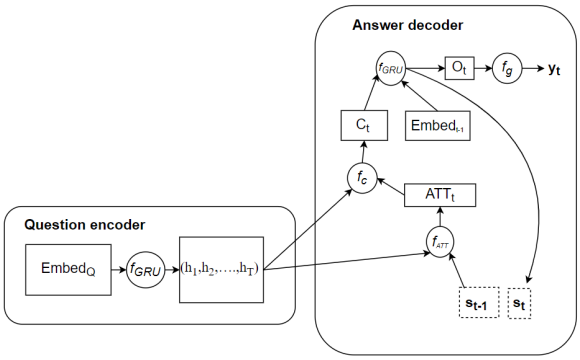

FIGURE 7. STL model.

iv) MTL-BC-CAM: The MTL-BC-CAM model is a modified version of MTL-BC where CAM is utilized for attention computation during answer generation. **This interim model is proposed to study the effectiveness of CAM to address the language model influence issue**. This model is trained using Algorithm 2. It utilizes a fixed task loss weight scheme for multi-task loss calculation during training, whereby the value of 0.1 is assigned as the binary classification task loss weight. The architecture of the

TABLE 8. Explanations on notation for the STL model.

Notation	Meaning
f_{GRU}	GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size densed vectors) [37]
$Embed_{t-1}$	Previously generated word embedding (at timestep t-1)
$Embed_Q$	Question embedding
$\mathbf{h_1},\mathbf{h_2},...,\mathbf{h_T}$	Hidden states of the question encoder
$\mathbf{s_t}$	Hidden states of the decoder at timestep t
f_{ATT}	Attention weight computation function
f_c	Context vector computation function
$\mathbf{ATT_t}$	Attention weights at timestep t
$\mathbf{C_t}$	Question context vector
$\mathbf{O_t}$	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t

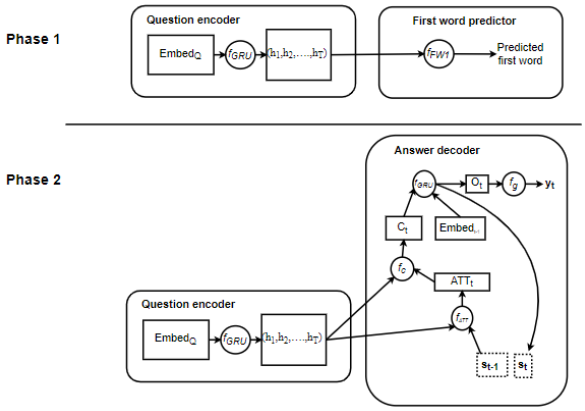

FIGURE 8. MTL-LTS model.

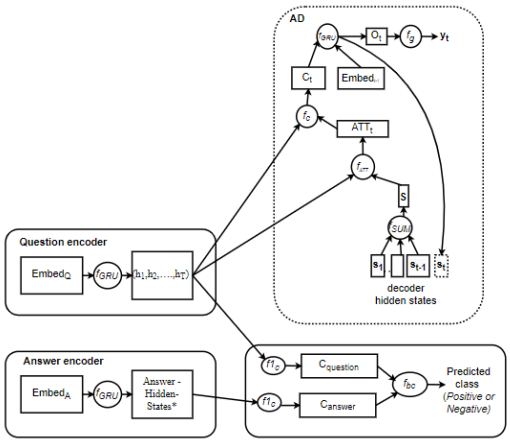

FIGURE 9. MTL-BC-CAM model.

MTL-BC-CAM model is illustrated in Figure 9. The notations are encapsulated in Table 10.

TABLE 9. Explanations on notation for the MTL-LTS model.

Notation	Meaning
f_{GRU}	GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size densed vectors) [37]
$Embed_{t-1}$	Previously generated word embedding (at timestep t−1)
$Embed_Q$	Question embedding
$\mathbf{h_1},\mathbf{h_2},...,\mathbf{h_T}$	Hidden states of the question encoder
f_{FW1}	First-word prediction function
$\mathbf{s_t}$	Hidden states of the decoder at timestep t
f_{ATT}	Attention weight computation function
f_c	Context vector computation function
$\mathbf{ATT_t}$	Attention weights at timestep t
$\mathbf{C_t}$	Question context vector
$\mathbf{O_t}$	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t

TABLE 10. Explanations on notation for the MTL-BC-CAM model.

Notation	Meaning
f_{GRU}	GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size densed vectors) [37]
$Embed_{t-1}$	Previously generated word embedding (at timestep t−1)
$Embed_Q$	Question embedding
$Embed_A$	Answer embedding
$\mathbf{h_1},\mathbf{h_2},...,\mathbf{h_T}$	Hidden states of the question encoder
f_{SUM}	Matrix summation operation
$\mathbf{s_1},\mathbf{s_2},...,\mathbf{s_T}$	Hidden states of the decoder
f_{ATT}	Attention weight computation function
$f_c, f1_c$	Context vector computation functions
$\mathbf{ATT_t}$	Attention weights at timestep t
$\mathbf{C_t}$	Question context vector
$\mathbf{O_t}$	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t
f_{bc}	Binary classification function

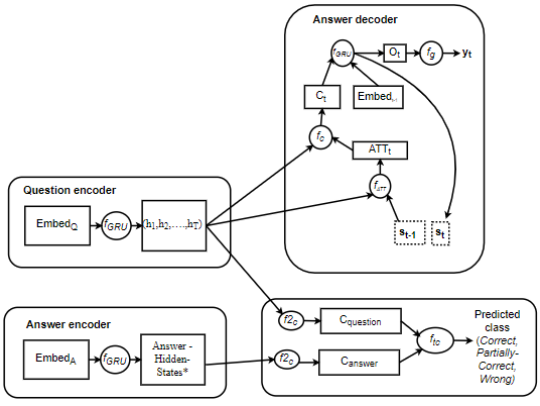

FIGURE 10. MTL-TC model.

v) The MTL-BC-DL model is proposed by modifying MTL-BC to utilize dynamic tasks loss weight

TABLE 11. Explanations on notation for the MTL-TC model.

Notation	Meaning
f_{GRU}	GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size densed vectors) [37]
$Embed_{t-1}$	Previously generated word embedding (at timestep t−1)
$Embed_Q$	Question embedding
$Embed_A$	Answer embedding
$\mathbf{h}_1,\mathbf{h}_2,...,\mathbf{h}_T$	Hidden states of the question encoder
$\mathbf{s}_t$	Hidden states of the decoder at timestep t
f_{ATT}	Attention weight computation function
f_c, $f1_c$	Context vector computation function
$\mathbf{ATT}_t$	Attention weights at timestep t
$\mathbf{C}_t$	Question context vector
$\mathbf{O}_t$	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t
f_{tc}	Ternary classification function

values for multi-task loss calculation during training. This model is trained using Algorithm 5. MTL-BC-DL shares the same architecture is as MTL-BC (Figure 6). **This interim model is proposed to study the effectiveness dynamic tasks loss weight scheme to reduce answer generation overfit.**

vi) The MTL-TC model is a modified version of MTL-BC that model that utilizes a TC as the auxiliary task. This model is trained using Algorithm 2. It utilizes a fixed-task loss weight scheme for multi-task loss calculation during training, whereby the value of 0.1 is assigned as the ternary classification task loss weight. The architecture of the MTL-TC model is shown in Figure 10. The notations are summarized in Table 11. **This interim model is proposed to study the effectiveness ternary classification to reduce question encoding overfit**.

vii) MTL-MFE: This model utilizes MFE, global attention mechanism during answer decoding and dynamic tasks loss weights for multi-task loss calculation. This model is trained using Algorithm 6. MTL-MFE architecture is illustrated in Figure 11. The notations are summarized in Table 12. **This interim model is proposed to study the effectiveness of MFE, which performs first-word and last-word predictions in parallel to reduce question encoding overfit**.

viii) SEQ2SEQ++: This is the ultimate model proposed to address all three issues, as described in the problem statement (language model influence issue, answer generation overfit issue, and question encoder overfit issue). SEQ2SEQ++ integrates all the newly proposed methods, namely, CAM, DL, TC, and MFE (Figure 5, Table 4). This model is trained using Algorithm 1.

Algorithm 3 STL Model Training

Input: Question (X)-Answer(Y) pairs, Maximum Answer Tokens to Generate (T), Number of Epochs

Steps:

For epoch 1 to Number of Epochs

For the batch of question-answer pairs, X and Y Do

1. Encoder: Perform question encoding, generate the hidden states for each timestep
2. Decoder:

2.1. Generate tokens with the highest probability one by one by feeding weighted hidden states from Encoder, *its own previously generated hidden state* and the ground-truth token until maximum answer length is reached, or end of sequence token is generated

2.2. Join all tokens to generate an answer (Y')

3. Calculate the cross-entropy loss (the difference between Y and Y')
4. Update the model parameters

End For

End For

Output: Trained Seq2Seq Model

TABLE 12. Explanations on notation for the MTL-MFE model.

Notation	Meaning
f_{GRU}	GRU function is a non-linear transformation which tansforms embedding into hidden states (fixed size densed vectors) [37]
$Embed_{t-1}$	Previously generated word embedding (at timestep t−1)
$Embed_Q$	Question embedding
$\mathbf{h}_1,\mathbf{h}_2,...,\mathbf{h}_T$	Hidden states of the question encoder
f_{FW}	First-word prediction function
f_{LW}	Last-word prediction function
$\mathbf{s}_t$	Hidden states of the decoder at timestep t
f_{ATT}	Attention weight computation function
f_c	Context vector computation function
$\mathbf{ATT}_t$	Attention weights at timestep t
$\mathbf{C}_t$	Question context vector
$\mathbf{O}_t$	Output (answer) context vector
f_g	Answer word prediction function
y_t	Predicted word at time step t

B. DATASETS

All models were trained and tested on two datasets. The first dataset, NarrativeQA [39] is a fiction-based dataset and was proposed for reading comprehension evaluation; however, we took the question and the first correct answer in our experiments because we only need that part. The second one,

Algorithm 4 MTL-LTS Training

Phase 1

Input: {Question (X), Answer(Y)} pairs, Maximum Epoch (E)

Steps:

Initialize Question Encoder, First-Word Predictor

Initialize loss $L_1 = 0$

For epoch 1 to Number of Epochs

ii) For batch 1 to Number of Batches Do

1.1 Perform question encoding: Generate the hidden states for each timestep

1.2 Perform first word prediction compute loss (L_1)

1.3 Update the model parameters (neural network weights)

End for Batch Looping

End for Epoch Looping

Output: Trained Learning-To-Start (*LTS*) model (first word prediction)

Phase 2

Input: Question (X)-Answer (Y) pairs, Maximum Answer Tokens to Generate (T), Number of Epochs

Steps:

Initialize Question Encoder, Answer Decoder

Initialize loss $L_2 = 0$

For epoch 1 to Number of Epochs

For the batch of question-answer pairs, X and Y

Do

1. Encoder: Perform question encoding, generate the hidden states for each timestep
2. Decoder:
   2.1 For $t > 1$, generate tokens with the highest probability one by one by feeding weighted hidden states from Encoder, its own previously generated hidden states and the ground-truth token until maximum answer length is reached, or end of sequence token is generated
   2.2 Join all tokens to generate an answer (Y')
3. Calculate the cross-entropy loss L_2 (the difference between Y and Y')
4. Update the model parameters

End For

End For

Output: Trained MTL-LTS Model

Algorithm 5 MTL-BC-DL Training

Input: {Question (X), Answer (Y), Label (L)} triplets, Maximum answer length (T), Maximum Epoch (E)

Steps:

Initialize Question Encoder, Answer Encoder, Answer Decoder and Binary Classifier

Initialize each loss $L_{MTL} = L_{ag} = L_{bc}$

Initialize each task loss weights $\alpha_{ag} = \alpha_{bc} = 0.5$

For epoch 1 to Number of Epochs

1. For batch 1 to Number of Batches Do
   1.1 Perform question encoding
   1.2 Perform answer encoding
   1.3 Perform binary classification and compute binary classification loss (L_{bc})
   1.4 Perform answer generation using the weighted hidden states from Encoder and *its own previously generated hidden state* and compute answer generation loss (L_{ag})
   1.5 Compute multi-task loss:

$$L_{MTL} = \alpha_{ag} * L_{ag} + \alpha_{bc} * L_{bc}$$

   1.6 Update the model parameters (neural network weights)
   1.7 For each task, $tk \subseteq \{ag, bc\}$: Calculate the average loss ($L_{tk\text{-}avg}$)
   End For Batch Looping
2. Calculate the total average loss for all tasks:

$$L_{avg\text{-}total} = L_{ag\text{-}avg} + L_{bc\text{-}avg}$$

3. For each task, $tk \subseteq \{ag, bc\}$
   3.1 Calculate new task loss weight:

$$\alpha_{tk} = L_{tk\text{avg}}/(L_{avg\text{-}total})$$

End for Epoch Looping

the SQuAD dataset (version 1.0) was published in 2006 by Rajpurkar *et al.* [40]. SQuAD is a question–answer dataset based on Wikipedia articles. For our experiments, we selected from SQuAD the questions that have answers, and we only took the first answer. Table 13 presents the details of both the datasets used in this experiment.

During training, the pairs of {questions, answers} are used by all models to learn to generate answers. For example (Table 14), for the question "How are plants different from animals?" the model will learn to produce the answer "Primary cell wall composed of the polysaccharides cellulose."

In addition to the answer-generation task, MTL-BC, MTL-BC-CAM, MTL-BC-DL, MTL-TC, and SEQ2SEQ++ require additional data to perform the answer classification.

MTL-BC, MTL-BC-CAM, and MTL-BC-DL require triplets of {question, correct answer, wrong answer} for answer-classification training. Table 15 shows the sample training data for the MTL-BC, MTL-BC-CAM, and MTL-BC-DL. The training dataset was generated as described in [18].

MTL-TC and SEQ2SEQ++ require quadruples of {questions, correct answers, partially correct answers, wrong answers} for answer classification training. Table 16 summarizes the sample training data for MTL-TC and SEQ2SEQ++. The training dataset was generated using the following approach.

Algorithm 6 MTL-MFE Training

Input: {Question (X), Answer(Y), First Word (FW), Last Word (LW)}, Maximum answer length (T), Maximum Epoch (E)

Steps:

Initialize Multifunctional Encoder (MFE), Answer Decoder (AD)

Initialize each loss $L_{MTL} = L_{ag} = L_{fw} = L_{lw} = 0$

Initialize each task loss weights $\alpha_{ag} = 0.34$, $\alpha_{fw} = \alpha_{lw} = 0.33$

For epoch 1 to Number of Epochs

1. For batch 1 to Number of Batches Do
   - 1.1 Perform question encoding
   - 1.2 Predict First Word and compute the first-word prediction losses (L_{fw})
   - 1.3 Predict Last Word and compute the last-word prediction losses (L_{fw})
   - 1.4 Perform answer generation and compute answer generation loss (L_{ag})
   - 1.5 Compute multi-task loss (L_{MTL}):

$$L_{MTL} = \alpha_{ag} * L_{ag} + \alpha_{fw} * L_{fw} + \alpha_{lw} * L_{lw}$$

   - 1.6 Update the model parameters (neural network weights)
   - 1.7 For each task, $tk \subseteq \{ag, fw, lw\}$: Calculate the average loss ($L_{tk\text{-}avg}$)

End for Batch Looping

2. Calculate the total average loss for all tasks: $L_{avg\text{-}total} = L_{fw\text{-}avg} + L_{lw\text{-}avg} + L_{ag\text{-}avg}$
3. For each task, tk $\subseteq$ {ag, fw, lw}
   - 3.1 Calculate new task loss weight:

$$\alpha_{tk} = L_{tk\text{-}avg}/(L_{avg\text{-}total})$$

End for Epoch Looping

Output: Trained MTL-MFE model

i) The original answer (gold answer) to the question which represents the correct answer is labeled as "Correct." It had a BLEU score of 1.

ii) The original answer is then manipulated by removing or adding one or more words to obtain a BLEU score of more than 0 and less than 1 as compared to the gold answer and labeled as "Partially Correct."

iii) Then, to find the third class to represent the wrong (negative) answer, any answer that belongs to another question in the dataset and has a BLEU score of 0 as compared to the gold answer, is randomly selected and labeled "Wrong."

C. EVALUATION METRICS

A total of three (3) evaluation metrics were used in this study to measure the performance of each model from various angles. They are: -

TABLE 13. Dataset details.

Dataset Name	NarrativeQA	SQuAD
Dataset summary	NarrativeQA is a fiction-based dataset and consists of proper English words.	SQuAD is a crowdsourced dataset based on Wikipedia articles.
Training (# of question-answer pairs)	24,000	24,819
Validation (# of question-answering pairs)	4,800	4,800
Testing (# of question-answering pairs)	1,000	1,000
Question vocabulary size (# of words)	17,294	19,569
Answer vocabulary size (# of words)	18,830	19,935
Maximum question length	19	17
Maximum answer length	16	17

TABLE 14. Sample training data for answer generation.

Pair 1	
Question	What are sleepy hollow renowned for?
Answer	Ghosts and a haunting atmosphere
Pair 2	
Question	How are plants different from animals?
Answer	Primary cell wall composed of the polysaccharides cellulose

TABLE 15. Sample training data for MTL-BC.

Sample 1		**BLEU Score**	**Label**
Question	What are sleepy hollow renowned for?		
Correct answer (original answer)	Ghosts and a haunting atmosphere	1	Correct
Wrong answer	St. Katherine's parish in San Francisco	0	Wrong
Sample 2		**BLEU Score**	**Label**
Question	How are plants different from animals?		
Correct answer (original answer)	Primary cell wall composed of the polysaccharides cellulose	1	Correct
Wrong answer	Global distillation	0	Wrong

1) BILINGUAL EVALUATION UNDERSTUDY (BLEU)

The BLEU metric [49] is the most popular metrics utilized in answer generation works such as but not limited to [18],

TABLE 16. Sample training data for SEQ2SEQ++.

Sample 1		**BLEU Score**	**Label**
Question	What are sleepy hollow renowned for?		
Correct answer (original answer)	Ghosts and a haunting atmosphere	1	Correct
Modified correct answer	Haunting atmosphere	0.23	Partially Correct
Wrong answer	St, Katherine's parish in San Francisco	0	Wrong
Sample 2		**BLEU Score**	**Label**
Question	How are plants different from animals?		
Correct answer (original answer)	Primary cell wall composed of the polysaccharides cellulose	1	Correct
Modified correct answer	of the polysaccharides cellulose	0.37	Partially Correct
Wrong answer	Global distillation	0	Wrong

[22], [34], [50], [51] and utilized in [18], one of our key reference works.It is used to measure the quality of machine translation versus human translation. It calculates an N-gram precision between the two sequences and imposes a commensurate penalty for a machine sequence that is shorter than the human sequence. **In addition to its original purpose, BLEU is used extensively for other text generation evaluations, including natural answer generation (NAG), and is reported to have a high correlation with human judgments of quality** [49]. Here, BLEU-2 corresponding to bi-gram versions of the approach is used to evaluate the generated answer versus the gold answer. **The BLEU metric measures the correctness of the systems being evaluated and gives a score between 0 and 1.** The higher the BLEU rate, the better is the model. In other words, a higher BLEU score indicates the generation of answers that are relevant to the question.

2) WORD ERROR RATE

Word error rate (WER) [52] measures the "mistakes" of the model. It **measures the rate of wrongly generated words** against the overall generated word. It gives a score of 0–1. A lower score indicates fewer errors. In other words, a low WER score indicates the generation of fewer high-frequency and generic words. The formula is given as (26).

WER complements BLEU; we added it to obtain a more holistic measurement of each model's performance.

$$WER = \sum WC_{wrong} / \sum WC_{total}, \tag{26}$$

where WC_{wrong} is wrongly generated word count, WC_{total} is total generated word count

TABLE 17. Generic experiment settings.

Parameter	**Value – fixed for all models/experiments**
Language version	Python 3.6
Key library version	Tensorflow 1.15.0
Environment	Google Colab Jupyter notebook environment with GPUs
Number of epochs	250 epochs
Training batch size	32 pairs
Neural network size	256 unit

3) DISTINCT-2

In addition to evaluating the correctness of the model using BLEU and error rate using WER metrics, the diversity of the answers generated by each model was also measured using a technique proposed in [53] and utilized in our key reference work [18]. Here, the Distinct-2 metric is used to measure diversity. Distinct-2 is the number of distinct bigrams divided by the total number of bigrams generated by the model. It has a value between 0 and 1. The higher the score, the more diverse the answers. In other words, generating more diverse answers indicates the generation of less high-frequency and generic answers. The formula is given as (27).

$$Distinct\text{-}2 = \sum Unique\ Bigrams / \sum Bigrams \tag{27}$$

D. EXPERIMENTAL SETTINGS

All models were implemented using Python version 3.6, and TensorFlow [54] version 1.15.0, which provides a serverless Jupyter notebook environment with GPUs for interactive development [55]. Each model was run for a maximum of 250 epochs and batches of 32 training pairs. The checkpoint with the lowest validation loss was used for the testing (experiments). For all models, the diverse beam search technique proposed in [56] was implemented during testing. Several combinations of group size and beam size were used, and the best outcome for each model is taken for performance comparison. Table 17 summarizes the settings.

Additionally, model-specific settings were used for the experiments. There are as follows:

i) MTL-BC [18] performs answer generation using the global attention mechanism and binary answer classification, and fixed weights are used to calculate the MTL loss during training.
ii) STL [41] is a single-task model, and it only performs an answer generation task by utilizing the global attention mechanism
iii) MTL-LTS [42] performs first-word prediction and then answer generation using the global attention mechanism.

iv) MTL-BC-CAM performs answer generation using the CAM method and binary answer classification, and fixed weights are used to calculate the MTL loss during training.
v) MTL-BC-DL performs answer generation using the global attention mechanism and binary answer classification. It utilizes dynamic tasks loss weight calculations during model training.
vi) MTL-TC performs answer generation using the global attention mechanism and ternary answer classification, and fixed weights are used to calculate the MTL loss during training.
vii) MTL-MFE performs answer generation using the global attention mechanism and first- and last-word predictions. It utilizes dynamic tasks loss weight calculations during model training
viii) SEQ2SEQ++ performs answer generation using CAM, first- and last-word predictions, and ternary answer classification. It utilizes dynamic tasks loss weight calculations during model training.

Table 18 presents a succinct summary of the model-specific settings.

TABLE 18. Experimented models and their settings.

Model	Category	Tasks				MTL Loss Calculation		Attention Mechanism	
		AG	AC	FW	LW	F	D	G	C
STL	BM	✓						✓	
MTL-BC	BM	✓	✓			✓		✓	
MTL-BC-CAM	IM	✓	✓			✓			✓
MTL-BC-DL	IM	✓	✓				✓	✓	
MTL-TC	IM	✓	✓			✓		✓	
MTL-MFE	IM	✓		✓	✓		✓	✓	
MTL-LTS	BM	✓		✓				✓	
SEQ2SEQ++	FM	✓	✓	✓	✓		✓		✓

Legend:

BM	Benchmark Model	IM	Interim Model
FM	Final model		
AG	Answer generation	FW	First-word prediction
AC	Answer classification	LW	Last-word prediction
F	Fixed	D	Dynamic
G	Global Attention Mechanism	C	Comprehensive attention mechanism

E. EXPERIMENTS RESULTS AND ANALYSIS

This section focuses into the experiment settings and analysis of the results based on (i) the comparison of interim models (MTL-BC-CAM, MTL-BD-DL, MTL-TC, and MTL-MFE) versus the benchmark models (STL and MTL-BC), and (ii) SEQ2SEQ++ against the benchmark models (STL and MTL-BC). The NarrativeQA and SQuaD datasets, and BLEU, WER and Distinct-2 metrics are used. We also report the analysis, signifance testing and case study for each issue.

1) INTERIM MODELS VERSUS BENCHMARK MODELS

This section presents and analyzes the experimental results of our interim models against specific benchmark models for each issue (language model influence, answer generation overfit, question encoder overfit and MTL-MFE versus MTL-LTS). These interim models were developed and tested to gauge the effectiveness of each of our proposed methods individually in addressing a specific issue as discussed in more details in the following sections.

a: LANGUAGE MODEL INFLUENCE

To understand the effectiveness of CAM versus the global attention mechanism for answer decoding, we compare MTL-BC-CAM with MTL-BC. Tables 19 and 20 show the experiment result for MTL-BC-CAM versus MTL-BC.

TABLE 19. MTL-BC-CAM vs MTL-BC.

Metric	Model	NarrativeQA	SQuAD
BLEU	MTL-BC	0.5709	0.6769
	MTL-BC-CAM	**0.6390**	**0.6888**
WER	MTL-BC	0.3492	0.2893
	MTL-BC-CAM	**0.2839**	**0.2622**
Distinct-2	MTL-BC	**0.8078**	0.8907
	MTL-BC-CAM	0.79	**0.8956**

Note
- For BLEU and Distinct-2, higher score means better performance
- For WER, lower score means better performance

NarrativeQA DATASET

For the BLEU metric, MTL-BC-CAM scored 0.6390 which is 11.9% higher than MTL-BC's score of 0.5709. As for WER, MTL-BC-CAM scored 0.2839 which is 18.7% lower than MTL-BC which scored 0.3492. However, for Distinct-2, MTL-BC-CAM scored 0.79 which is lower by 2.3% than MTL-BC's score of 0.8078. This experiment on NarrativeQA shows a mixed result whereby MTL-BC-CAM performed better than MTL-BC in two (2) metrics which are BLEU and WER. MTL-BC-CAM performed worse than MTL-BC in terms of Distinct-2 score. This means MTL-BC-CAM can produce answers with higher correctness (represented by a higher BLEU score) and lower errors

TABLE 20. Percentage improvements MTL-BC-CAM vs MTL-BC.

Dataset	BLEU	WER	Distinct-2
NarrativeQA	11.9%	18.7%	-2.3% *
SQuAD	1.8%	9.4%	0.6%

Note
- * Negative values indicate MTL-BC performed better than MTL-BC-CAM

(represented by a lower WER score). However, in terms of diversity MTL-BC performed better than MTL-BC-CAM which is represented by a higher Distinct-2 score.

SQuaD DATASET

For the BLEU metric, MTL-BC-CAM scored 0.6888 which is 1.8% higher than MTL-BC's score of 0.6769. As for WER, MTL-BC-CAM scored 0.2622 which is 9.4% lower than MTL-BC which scored 0.2893. For the Distinct-2 metric, MTL-BC-CAM scored 0.8956 which is 0.6% higher than MTL-BC's score which is 0.8907. This experiment result on the SQuaD dataset shows MTL-BC-CAM performed better than MTL-BC in all the metrics. This means MTL-BC-CAM can produce answers with higher correctness (represented by higher BLEU score), lower errors (represented by lower WER score), and higher diversity (represented by higher Distinct-2 scores).

ANALYSIS

MTL-BC-CAM scored higher than MTL-BC in two (2) metrics for NarrativeQA dataset (BLEU: 11.9% and WER: 18.7%) and in all the metrics for SQuAD dataset (BLEU: 1.8%, WER: 9.4% and Distinct-2: 0.6%). This experiment outcome shows that by utilizing CAM, MTL-BC-CAM can capture the representation of the answer more precisely without loss of important information which is the question and generated answer words during each decoding step. Thus, CAM is a more effective attention mechanism than the global attention mechanism to address the language model influence issue. This demonstrates that answer generation is more effective by utilizing all the decoder's previously generated hidden states which represent the decoder's generated words.

SIGNIFICANCE

To evaluate whether the performance of MTL-BC-CAM versus MTL-BC scores is statistically significant or not, the paired Student's t-test statistical tests were performed on the BLEU and WER scores for both datasets. Results (Table 21) indicate that MTL-BC-CAM performed significantly better than MTL-BC on three (3) measurements which are BLEU-NarrativeQA, WER-NarrativeQA, and WER-SQuaD. BLEU-SQuaD measurement shows an insignificant difference. Student's t-test could not be performed for Distinct-2, as Distinct-2 is an overall score of model diversity and is not based on individual answers generated.

TABLE 21. Significance test MTL-BC-CAM versus MTL-BC.

Measurement	MTL-BC-CAM	MTL-BC	Statistics	Significance
BLEU - NarrativeQA N=1,000	Mean = 0.6390 SD = 0.3845	Mean = 0.5709 SD = 0.4133	t = 4.9322 p = less than 0.0001	Significant
BLEU-SQuaD N=1,000	Mean = 0.6888 SD = 0.4252	Mean = 0.6769 SD = 0.4320	t = 0.9588 p = 0.3379	Not significant
WER-NarrativeQA N=1,000	Mean = 0.2839 SD = 0.3556	Mean = 0.3492 SD = 0.3900	t = 5.0116 p = less than 0.0001	Significant
WER-SQuaD N=1,000	Mean = 0.2622 SD = 0.4085	Mean = 0.2893 SD = 0.4230	t = 2.2998 p = 0.0217	Significant

Note:
N: number of samples (test questions)
Mean: average scores for that category
SD: standard deviation of the scores for that category
t: t-score
p: p-score. p-score < 0.05 indicates significant difference between the two models.

CASE STUDY

Table 22 shows two sample questions and the corresponding generated answers by each of the experimented models. Column "BLEU score" shows the BLEU score of the respective answers generated by each model. Column "Frequency of term/phrase" shows the frequency of selected words in the respective datasets.

Sample 1 output shows that MTL-BC failed to generate the correct answer because halfway through the answer generation, it generated the word "plans" which has a frequency of 36 instead of "patient" which has a frequency of 18. For Sample 2, MTL-BC generated the end token ("⟨end⟩") too soon. The end token has a very high frequency of 24,819 compared to "of" which is 4078.

The result shows that MTL-BC-CAM can generate correct answers for both questions even though MTL-BC failed to generate the correct answers.

b: ANSWER GENERATION OVERFIT

To understand the effectiveness of the dynamic-task loss weights scheme versus the fixed-task loss weights scheme in reducing answer generation overfit, we compare MTL-BC-DL with MTL-BC. Tables 23 and 24 show the experiment result for MTL-BC-DL versus MTL-BC using the NarrativeQA and SQuaD dataset.

NarrativeQA DATASET

For the BLEU metric, MTL-BC-DL scored 0.5979 which is 4.7% higher than MTL-BC's score of 0.5709. As for WER,

TABLE 22. Sample output MTL-BC-CAM versus MTL-BC.

Sample 1	Dataset: NarrativeQA	BLEU score	Frequency of term/phrase
Question	Why did Capa stop seeing the color red?		
Gold answer	He suffered from psychosomatic color blindness because of seeing his patient murdered		from:808 patient:18 murdered:61
Answer generated by MTL-BC	He suffered from psychosomatic color blindness because of seeing his ***plans on*** his patient <end>	0.74	*plans*:36 *on*:6840
Answer generated by MTL-BC-CAM	He suffered from psychosomatic color blindness because of seeing his patient murdered <end>	1	
Sample 2	**Dataset: SQuAD**	**BLEU score**	**Frequency of term/phrase**
Question	Who was invested as the leader of the Greek Cypriot population ?		
Gold answer	Head of the church of Cyprus		*church*:119 *of*:4078
Answer generated by MTL-BC	Head of the church ***<end>***	0.61	*<end>*:24819
Answer generated by MTL-BC-CAM	Head of the church of Cyprus <end>	1	

*General Note: All generated sentences will have <end> which symbolizes end of sentence, but it will not be shown to the user.

frequency of 24,819 compared to "of" which is 4078.

MTL-BC-DL scored 0.3183, which is 8.9% lower than MTL-BC which scored 0.3492. For the Distinct-2 metric, MTL-BC scored slightly higher (0.6%) than MTL-BC-DL. MTL-BC's score was 0.8078 against MTL-BC-DL's score of 0.8029. This experimental result on the NarrativeQA dataset confirms that MTL-BC-DL performed better than MTL-BC in the BLEU and WER metrics. MTL-BC fared better than MTL-BC-DL. This means MTL-BC-DL can produce answers with higher correctness (represented by a higher BLEU score) and lower errors (represented by a lower WER score). On the other hand, MTL-BC generated a higher diversity (represented by higher Distinct-2 scores).

SQuaD DATASET

For the BLEU metric, MTL-BC-DL scored 0.6878 which is 1.6% higher than MTL-BC-DL's score of 0.6769. As for the WER metric, MTL-BC-DL scored 0.2807 which is 3.0% lower than MTL-BC which scored 0.2893. For the Distinct-2 metric, MTL-BC scored slightly higher (0.7%) than MTL-BC-DL. MTL-BC's score was 0.8907 against MTL-BC-DL's score of 0.8842. This experiment results on the SQuaD dataset show MTL-BC-DL performed better than MTL-BC in the BLEU and WER metrics. MTL-BC fared better than MTL-BC-DL. This is a similar result to the experiment on the NarrativeQA dataset. The result shows that MTL-BC-DL can

TABLE 23. MTL-BC-DL vs MTL-BC.

Metric	Model	NarrativeQA	SQuAD
BLEU	MTL-BC	0.5709	0.6769
	MTL-BC-DL	**0.5979**	**0.6878**
WER	MTL-BC	0.3492	0.2893
	MTL-BC-DL	**0.3183**	**0.2807**
Distinct-2	MTL-BC	**0.8078**	**0.8907**
	MTL-BC-DL	0.8029	0.8842

Note
- For BLEU and Distinct-2, higher score means better performance
- For WER, lower score means better performance

TABLE 24. Percentage improvements MTL-BC-DL vs MTL-BC.

Dataset	BLEU	WER	Distinct-2
NarrativeQA	4.7%	8.9%	-0.6% *
SQuAD	1.6%	3.0%	-0.7%

* Negative values indicate MTL-BC performed better than MTL-BC-DL

TABLE 25. Significance test MTL-BC-DL versus MTL-BC.

Measurement	MTL-BC-DL	MTL-BC	Statistics	Significance
BLEU - Narrative QA N=1000	Mean = 0. 5979 SD = 0. 4013	Mean = 0.5709 SD = 0.4133	t = 1.8870 p = 0.0595	Not significant
BLEU-SQuaD N=1000	Mean = 0. 6878 SD = 0. 4286	Mean = 0.6769 SD = 0.4320	t = 0.9146 p = 0.3606	Not significant
WER-Narrative QA N=1000	Mean = 0. 3183 SD = 0. 3788	Mean = 0.3492 SD = 0.3900	t = 2.3301 p = 0.0200	Significant
WER-SQuaD N=1000	Mean = 0. 2807 SD = 0. 4169	Mean = 0.2893 SD = 0.4230	t = 0.7324 p = 0.4641	Not significant

Note:
N: number of samples (test questions)
Mean: average scores for that category
SD: standard deviation of the scores for that category
t: t-score
p: p-score. p-score < 0.05 indicates significant difference between the two models.

produce answers with higher correctness (represented by a higher BLEU score) and lower errors (represented by a lower WER score). On the other hand, MTL-BC produced higher diversity (represented by higher Distinct-2 scores).

ANALYSIS

MTL-BC-DL scored higher than MTL-BC in two (2) metrics for both NarrativeQA (BLEU: 4.7% and WER: 8.9%) and SQuAD datasets (BLEU: 1.6% and WER: 3.0%) respectively. This is because MTL-BC-DL utilizes the Dynamic Tasks Loss Weights Scheme (DL) which recalculates each of the task's loss weights during each epoch and assigns the new values to be used for the next epoch. A task's loss represents the difference between the predicted value against the actual value. Higher the loss means the task is not performing well relatively and has to do a lot more learning to improve its prediction to reduce the loss. A relatively higher task loss weight for a task means the task contributes more to the MTL loss. A higher loss will ensure model learning continues and doesn't stop early. There will be bigger updates to the neural network weights accordingly to ensure the model can predict better in future epochs. Thus, answer generation overfit issue which occurs in fixed tasks loss weight scheme can be avoided or reduced effectively by utilizing DL.

SIGNIFICANCE

To evaluate whether the performance of MTL-BC-DL versus MTL-BC scores is statistically significant or not, the paired Student's t-test statistical tests were performed on the BLEU and WER scores for both datasets. Results (Table 25) indicate that MTL-BC-DL performed significantly better than MTL-BC on the WER-NarrativeQA measurement. Other measurements show insignificant differences.

CASE STUDY

Table 26 shows two (2) sample questions and the corresponding generated answers by each of the experimented models. Column "BLEU score" shows the BLEU score of the respective answers generated by each model. Column "Frequency of term/phrase" shows the frequency of selected words in the respective datasets.

Sample 1 output shows that MTL-BC generated the word "father" which has a higher frequency of 359 instead of the word "cousins" with a frequency of 31. For Sample 2, MTL-BC generated the common word "the" which has a very high frequency of 7479. The correct word that should be generated is "battle." question. In both cases, MTL-BC-DL can generate the correct answer. This outcome shows that by utilizing the dynamic tasks loss weight scheme, the performance of an MTL model can be further improved.

c: QUESTION ENCODER OVERFIT

To understand the effectiveness of our proposed methods in reducing the question encoder overfit against the benchmark models, we compare MTL-TC with MTL-BC and MTL-MFE with MTL-LTS.

MTL-TC Versus MTL-BC: Tables 27 and 28 show the experiment result of MTL-TC and MTL-BC on NarrativeQA and SQuAD datasets.

TABLE 26. Sample output MTL-BC-DL versus MTL-BC.

Sample 1	**Dataset: NarrativeQA**	**BLEU score**	**Frequency of term/phrase**
Question	how are benjamin and flopsy related ?		
Gold answer	they are ***cousins***		*cousins*: 31
Answer generated by MTL-BC	they are ***father*** <*end*>	0.63	*father*: 359
Answer generated by MTL-BC-DL	they are ***cousins*** <*end*>	1	
Sample 2	**Dataset: SQuAD**	**BLEU score**	**Frequency of term/phrase**
Question	what defeat led to prussia having to swear its allegiance to napoleon ?		
Gold answer	***battle*** of jena-auerstedt		*battle*: 72
Answer generated by MTL-BC	***the*** feast of jena-auerstedt <end>	0.48	*the*: 7479
Answer generated by MTL-BC-DL	***battle*** of jena-auerstedt <end>	1	

*General Note: All generated sentences will have <end> which symbolizes end of sentence, but it will not be shown to the user.

TABLE 27. MTL-TC vs MTL-BC.

Metric	**Model**	**NarrativeQA**	**SQuAD**
BLEU	MTL-BC	0.5709	0.6769
	MTL-TC	**0.6880**	**0.6954**
WER	MTL-BC	0.3492	0.2893
	MTL-TC	**0.2482**	**0.2720**
Distinct-2	MTL-BC	0.8078	**0.8907**
	MTL-TC	**0.815**	0.8906

Note

- For BLEU and Distinct-2, higher score means better performance
- For WER, lower score means better performance

TABLE 28. Percentage improvements MTL-TC vs MTL-BC.

Dataset	**BLEU**	**WER**	**Distinct-2**
NarrativeQA	20.5%	28.9%	0.9%
SQuAD	2.7%	6%	-0.01%

*Negative values indicate MTL-BC performed better than MTL-TC

TABLE 29. Significance test MTL-TC versus MTL-BC.

Measurement	MTL-TC	MTL-BC	Statistics	Significance
BLEU - Narrative QA N=1000	Mean = 0.6880 SD = 0.3937	Mean = 0.5709 SD = 0.4133	t = 8.5515 p = less than 0.0001	Significant
BLEU-SQuaD N=1000	Mean = 0.6954 SD = 0.4275	Mean = 0.6769 SD = 0.4320	t = 1.6113 p = 0.1074	Not significant
WER-Narrative QA N=1000	Mean = 0.2482 SD = 0.3624	Mean = 0.3492 SD = 0.3900	t = 7.9536 p = less than 0.0001	Significant
WER-SQuaD N=1000	Mean = 0.2720 SD = 0.4154	Mean = 0.2893 SD = 0.4230	t = 1.5665 p = 0.1175	Not significant

Note:
N: number of samples (test questions)
Mean: average scores for that category
SD: standard deviation of the scores for that category
t: t-score
p: p-score. p-score < 0.05 indicates significant difference between the two models.

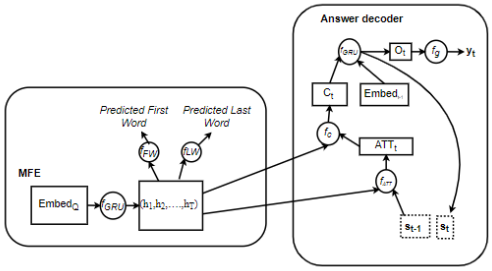

FIGURE 11. MTL-MFE model.

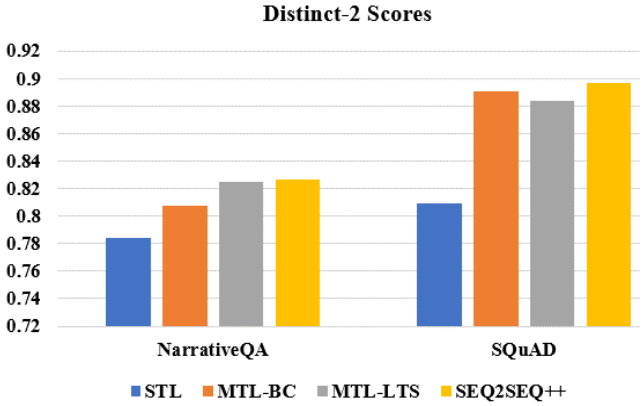

FIGURE 12. BLEU scores.

NarrativeQA DATASET

For the BLEU metric, MTL-TC scored 20.5% higher than MTL-BC. MTL-TC and MTL-BC scored 0.0.6880 and 0.5709 respectively. As for the WER metric, MTL-TC scored 0.2482 which is 28.9% lower than MTL-BC which scored 0.3492. For the Distinct-2 metric, MTL-TC scored more than MTL-BC. MTL-TC's score was 0.815 against MTL-BC's score of 0.8078. This experiment result on the NarrativeQA dataset shows MTL-TC performed better than MTL-BC in all the evaluation metrics. This means MTL-TC can produce answers with higher correctness (represented by higher BLEU score), lower errors (represented by lower WER score), and higher diversity (represented by higher Distinct-2 scores) than MTL-BC.

SQuaD DATASET

For the BLEU metric, MTL-TC scored 0.6954 which is 2.7% higher than MTL-BC's score of 0.6769. As for the WER metric, MTL-TC scored 0.2720 which is 6% lower than MTL-BC's score of 0.2893. For the Distinct-2 metric, MTL-BC and MTL-TC's scores were almost the same with only a very marginal difference of 0.01%. This experiment result on the SQuAD dataset shows MTL-TC performed better than MTL-BC in all the BLEU and WER metrics. This means MTL-TC can produce answers with higher correctness (represented by a higher BLEU score) and lower errors (represented by a lower WER score) than MTL-BC.

ANALYSIS

MTL-TC scored higher than MTL-BC in all the metrics for NarrativeQA dataset (BLEU: 20.5%, WER: 28.9% and Distinct-2: 0.9%) and two (2) metrics for SQuAD datasets (BLEU: 2.7%, and WER: 6.0%). This experiment outcome shows that by utilizing a slightly more complex task (ternary classification) as compared to binary classification, the question encoder needs to fine-tune its encoding to ensure the receiving networks can perform their tasks well.

SIGNIFICANCE

To evaluate whether the performance of MTL-TC versus MTL-BC scores is statistically significant or not, the paired Student's t-test statistical tests were performed on the BLEU and WER scores for both datasets. Results (Table 29) indicate that MTL-TC performed significantly better than MTL-BC in two (2) measurements which are BLEU-NarrativeQA and WER-NarrativeQA. Other measurements showed insignificant differences.

In this case, the question encoding is passed to the answer decoder and ternary classifier. This demonstrates that answer generation is more effective when question encoding overfit can be reduced by utilizing a slightly more complex question-answer classification method.

CASE STUDY

Table 30 shows two (2) sample questions and the corresponding generated answers by each of the experimented models.

TABLE 30. Sample output MTL-TC versus MTL-BC.

Sample 1	**Dataset: NarrativeQA**	**BLEU score**	**Frequency of term/phrase**
Question	What are sleepy hollow renowned for?		
Gold answer	Ghosts and a haunting ***atmosphere***		*atmosphere*: 5
Answer generated by MTL-BC	Ghosts and a haunting ***women*** <end>	0.76	*women*: 61
Answer generated by MTL-TC	Ghosts and a haunting atmosphere <end>	1	
Sample 2	**Dataset: SQuAD**	**BLEU score**	**Frequency of term/phrase**
Question	What was the purpose of the thermionic triode?		
Gold answer	Amplified radio ***technology*** and long-distance telephony		*technology*: 34
Answer generated by MTL-BC	Amplified radio ***and*** long-distance telephony <end>	0.5	*and*: 3867
Answer generated by MTL-TC	Amplified radio technology and long-distance telephony <end>	1	

*General Note: All generated sentences will have <end> which symbolizes end of sentence, but it will not be shown to the user.

TABLE 31. MTL-MFE vs MTL-LTS.

Metric	**Model**	**NarrativeQA**	**SQuAD**
BLEU	MTL-LTS	0.5665	0.5288
	MTL-MFE	**0.7604**	**0.7708**
WER	MTL-LTS	0.3323	0.4353
	MTL-MFE	**0.1706**	**0.2069**
Distinct-2	MTL-LTS	0.8247	0.8837
	MTL-MFE	**0.8283**	**0.8939**

Note
- For BLEU and Distinct-2, higher score means better performance
- For WER, lower score means better performance

TABLE 32. Percentage improvements MTL-MFE vs MTL-LTS.

Dataset	**BLEU**	**WER**	**Distinct-2**
NarrativeQA	34.2%	48.7%	0.4%
SQuAD	45.8%	52.5%	1.2%

Column ''BLEU score'' shows the BLEU score of the respective answers generated by each model. Column ''Frequency of term/phrase'' shows the frequency of selected words in the respective dataset.

In Sample 1, an incorrect answer was generated by MTL-BC because it generated the word ''women'' which occurs 61 times in the dataset instead of the correct word ''atmosphere'' which occurs only 5 times in the dataset. Sample 2 output shows that MTL-BC generated the word ''and'' which has a very high frequency in the dataset which is 3867 instead of the correct word which is ''technology'' which has a frequency of 34 only. In both samples, MTL-TC can generate the correct answers. The MTL-TC which uses ternary classification tasks during training can generate correct answers as compared to when using a binary classification task. This shows by training an MTL model with ternary classification can reduce question encoding overfit and thus reducing the occurrence of frequently occurring words in answers.

d: MTL-MFE VERSUS MTL-LTS

Tables 31 and 32 show the experiment result of MTL-MFE and MTL-LTS on NarrativeQA and SQuAD datasets.

NarrativeQA DATASET

For the BLEU metric, MTL-MFE scored 34.2% higher than MTL-LTS. MTL-MFE and MTL-LTS scored 0.7604 and 0.5665 respectively. As for the WER metric, MTL-MFE scored 0.1706 which is 48.7% lower than MTL-LTS which scored 0.3323. For the Distinct-2 metric, MTL-MFE scored slightly higher (0.4%) than MTL-LTS. MTL-MFE's score was 0.8283 against MTL-LTS's score of 0.88247. This experiment result on the NarrativeQA dataset shows MTL-MFE performed very much better than MTL-LTS in all the evaluation metrics. This means MTL-MFE can produce answers with higher correctness (represented by higher BLEU score), lower errors (represented by lower WER score), and higher diversity (represented by higher Distinct-2 scores) than MTL-LTS.

SQuaD DATASET

For the BLEU metric, MTL-MFE scored 0.7708 which is 45.8% higher than MTL-LTS's score of 0.5288. As for the WER metric, MTL-MFE scored 0.2069 which is 52.5% lower than MTL-LTS's score of 0.4353. For the Distinct-2 metric, MTL-MFE scored higher (1.2%) than MTL-LTS. MTL-MFE's score was 0.8939 against MTL-BC-DL's score of 0.8837. This experiment result on the SQuAD dataset shows MTL-MFE performed very much better than MTL-LTS in all the evaluation metrics. This means MTL-MFE can produce answers with higher correctness (represented by higher BLEU score), lower errors (represented by lower WER score), and higher diversity (represented by higher Distinct-2 scores) than MTL-LTS.

ANALYSIS

MTL-MFE scored higher than MTL-LTS in all the metrics for both NarrativeQA dataset (BLEU: 34.2%,

TABLE 33. Significance test MTL-TC versus MTL-BC.

Measurement	MTL-MFE	MTL-LTS	Statistics	Significance
BLEU-NarrativeQA N=1000	Mean = 0.7604 SD = 0.3715	Mean = 0.5665 SD = 0.3928	t = 15.4329 p = Less than 0.0001	Significant
BLEU-SQuAD N=1000	Mean = 0.7708 SD = 0.3919	Mean = 0.5288 SD = 0.4597	t = 17.6718 p = Less than 0.0001	Significant
WER-NarrativeQA N=1000	Mean = 0.1706 SD = 0.3265	Mean = 0.3323 SD = 0.3605	t = 13.8294 p = Less than 0.0001	Significant
WER-SQuAD N=1000	Mean = 0.2069 SD = 0.3783	Mean = 0.4353 SD = 0.4587	t = 16.8577 p = Less than 0.0001	Significant

Note:
N: number of samples (test questions)
Mean: average scores for that category
SD: standard deviation of the scores for that category
t: t-score
p: p-score. p-score < 0.05 indicates significant difference between the two models.

TABLE 34. Sample output MTL-MFE versus MTL-LTS.

Sample 1	Dataset: NarrativeQA	BLEU score	Frequency of term/phrase
Question	What is the main conflict between Agellius and his family?		
Gold answer	They are pagans that want him to abandon ***Christianity***		*Christianity*: 11
Answer generated by MTL-LTS	They are pagans that want him to abandon ***her*** husband <end>	0.79	*her*: 3710
Answer generated by MTL-MFE	They are pagans that want him to abandon Christianity <end>	1	
Sample 2	**Dataset: SQuAD**	**BLEU score**	**Frequency of term/phrase**
Question	When can patents not be filed in most countries?		
Gold answer	After prior art has been made ***public***		*public* : 153
Answer generated by MTL-LTS	After prior art has been made ***<end>***	0.85	*<end>*: 24819
Answer generated by MTL-MFE	After prior art has been made public <end>	1	

*General Note: All generated sentences will have <end> which symbolizes end of sentence, but it will not be shown to the user.

WER: 48.7% and Distinct-2: 0.4%) and SQuAD datasets (BLEU: 45.8%, WER: 52.5% and Distinct-2: 1.2%). This experiment outcome shows that the MTL-MFE model which is trained in parallel mode is more effective to reduce question encoder overfit as compared to MTL-LTS training which is based on sequential mode training. By training in parallel mode, the question encoder needs to fine-tune its encoding to ensure all the receiving networks can perform their tasks well. In this case, the question encoding is passed to the answer decoder, first-word predictor, and last-word predictor. This demonstrates that answer generation is more effective when question encoding overfit can be reduced by utilizing a multi-functional encoder and performing training in parallel mode.

SIGNIFICANCE

To evaluate whether the performance of MTL-MFE versus MTL-LTS scores is statistically significant or not, the paired Student's t-test statistical tests were performed on the BLEU and WER scores for both datasets. Results (Table 33) indicate that MTL-MFE performed significantly better than MTL-LTS in all the measurements.

CASE STUDY

Table 34 shows two (2) sample questions and the corresponding generated answers by each of the experimented models. Column "BLEU score" shows the BLEU score of the respective answers generated by each model. Column "Frequency of term/phrase" shows the frequency of selected words in the respective dataset.

Sample 1 output shows that MTL-LTS generated the word "her" which has a very high-frequency count of 58,556 in the dataset instead of the word "Christianity" which has a frequency of 11 only. Similarly, for Sample 2, a shorter and incorrect answer was generated by MTL-LTS because it generated the end token ("⟨end⟩") too early. The end token has a very high frequency of 24819 compared to the word "public" which is only 153. The MTL-MFE can generate correct answers as compared to MTL-LTS. This shows by training the auxiliary tasks in parallel, question encoding overfitting can be reduced.

2) SEQ2SEQ++ VERSUS BENCHMARK MODELS

This section presents the analysis of improvements by SEQ2SEQ++ against the benchmark models (MTL-BC [18], STL [41], and MTL-LTS [42]) for each dataset utilized in this study. The results are presented in Tables 35 and 36.

a: EXPERIMENT ON NarrativeQA DATASET

For the BLEU metric (Figure 12), SEQ2SEQ++ scored the highest at 0.8245. This score is much higher than the benchmark models STL, MTL-BC, and MTL-LTS, which only scored 0.5399, 0.5709, and 0.5665, respectively. SEQ2SEQ++ scored 44.4% higher than MTL-BC, which is the next best model in terms of BLEU score for NarrativeQA.

As for the WER metric (Figure 13), SEQ2SEQ++ had the lowest score of 0.1368. This score is much lower

TABLE 35. SEQ2SEQ++ versus benchmark models.

Metric	Model	NarrativeQA	SQuAD
BLEU	STL	0.5399	0.5087
	MTL-BC	0.5709	0.6769
	MTL-LTS	0.5665	0.5288
	SEQ2SEQ++	**0.8245**	**0.7941**
WER	STL	0.3701	0.4145
	MTL-BC	0.3492	0.2893
	MTL-LTS	0.3323	0.4353
	SEQ2SEQ++	**0.1368**	**0.1815**
Distinct-2	STL	0.7838	0.809
	MTL-BC	0.8078	0.8907
	MTL-LTS	0.8247	0.8837
	SEQ2SEQ++	**0.8264**	**0.8972**

Note

- For BLEU and Distinct-2, higher score means better performance
- For WER, lower score means better performance

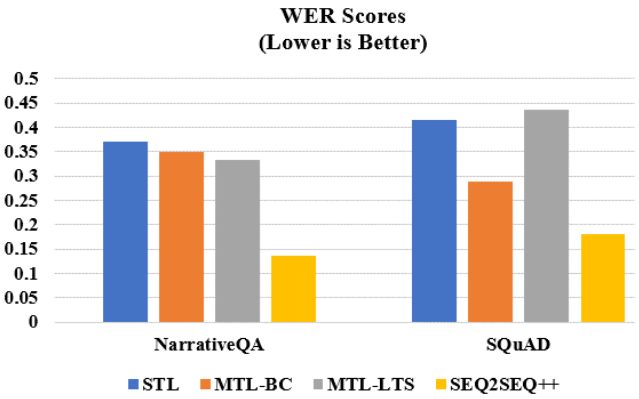

FIGURE 13. WER scores.

than the benchmark models STL, MTL-BC, and MTL-LTS, which scored 0.3701, 0.3492, and 0.3323, respectively. SEQ2SEQ++ scored 58.9% lower than MTL-LTS, which is the next best model in terms of the WER score for NarrativeQA.

For the Distinct-2 metric (Figure 14), SEQ2SEQ++ scored the highest with 0.8264, followed by MTL-LTS with 0.8247, MTL-BC with 0.8078, and STL with 0.7838. For this metric, the score of the SEQ2SEQ++ model is only slightly higher (0.2%) than the MTL-LTS score.

b: EXPERIMENT ON SQuaD DATASET

For the BLEU metric (Figure 12), SEQ2SEQ++ scored the highest at 0.7941. This score is much higher than

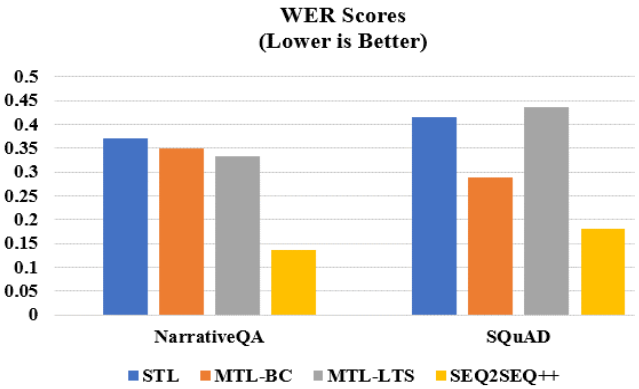

FIGURE 14. Distinct-2 scores.

TABLE 36. Percentage improvements SEQ2SEQ++ versus benchmark models.

Benchmark Model	Dataset	BLEU	WER	Distinct-2
STL	Narrative QA	52.71%	63.04%	5.44%
	SQuAD	56.1%	65.21%	10.9%
MTL-BC	Narrative QA	44.42%	60.82%	2.85%
	SQuAD	17.31%	37.26%	0.73%
MTL-LTS	Narrative QA	45.54%	58.83%	0.21%
	SQuAD	50.17%	58.3%	1.53%

the benchmark models STL, MTL-BC, and MTL-LTS, which only scored 0.5087, 0.6769, and 0.5288, respectively. SEQ2SEQ++ scored 17.3% higher than MTL-BC, which is the next best model in terms of BLEU score for the SquaD dataset.

As for the WER metric (Figure 13), SEQ2SEQ++ had the lowest score of 0.1815. This score is much lower than the benchmark models STL, MTL-BC, and MTL-LTS, which scored 0.4145, 0.2893, and 0.4353, respectively. SEQ2SEQ++ scored 37.3% lower than MTL-BC, which is the next best model in terms of the WER score for SquaD.

For the Distinct-2 metric (Figure 14), SEQ2SEQ++ scored the highest with 0.8972, followed by MTL-MC with 0.8907, MTL-LTS with 0.8837, and STL with 0.809. For this metric, the score of the SEQ2SEQ++ model is only slightly higher (0.7%) than the MTL-BC'S score.

c: ANALYSIS

SEQ2SEQ++ achieved the best performance (Figures 11,12 and 13) for all the evaluation metrics and for both datasets as compared to all the benchmark models STL, MTL-BC, and MTL-LTS. Table 36 shows the percentage difference of the SEQ2SEQ++ model against each benchmark model. This result indicates that SEQ2SEQ++ can address

TABLE 37. Significance test SEQ2SEQ++ versus benchmark models.

Measurement	SEQ2SEQ++	Second Best Model	Statistics	Significance
BLEU - Narrative QA N=1000	Mean = 0.8245 SD = 0.3235	MTL-BC Mean = 0.5709 SD = 0.4133	t = 19.8079 p = less than 0.0001	Significant
BLEU-SQuaD N=1000	Mean = 0.7941 SD = 0.3769	MTL-BC Mean = 0.6769 SD = 0.4320	t = 9.7203 p = less than 0.0001	Significant
WER-Narrative QA N=1000	Mean = 0.1368 SD = 0.2881	MTL-LTS Mean = 0.3323 SD = 0.3605	t = 18.2960 p = less than 0.0001	Significant
WER-SQuaD N=1000	Mean = 0.1815 SD = 0.3615	MTL-BC Mean = 0.2893 SD = 0.4230	t = 9.2909 p = less than 0.0001	Significant

Note:
N: number of samples (test questions)
Mean: average scores for that category
SD: standard deviation of the scores for that category
t: t-score
p: p-score. p-score < 0.05 indicates significant difference between the two models.

all three issues (language model influence, answer generation overfit, and question encoding overfit) more effectively than the benchmark models. It can generate answers with higher quality (highest BLEU score), lower error rate (lowest WER score), and higher diversity (highest Distinct-2 score) in comparison with all the other benchmark models.

3) SIGNIFICANCE

To evaluate whether the SEQ2SEQ++ model's scores against the second-best benchmark model are statistically significant, Student's t-test statistical tests were performed for BLEU and WER scores. The results (Table 37) indicate that the performance difference in terms of BLEU and WER scores of the SEQ2SEQ++ model against the next best models is statistically significant. Student's t-test could not be performed for Distinct-2, as Distinct-2 is an overall score of model diversity and is not based on individual answers generated.

4) CASE STUDY

This section studies samples of answers generated by each model as a case study to highlight the advantages of each of the novel methods and the SEQ2SEQ++ model in addressing the issues in Seq2Seq learning. One (1) sample is shown from each dataset. The column "Frequency of term/phrase" provides details on the frequency of selected words in the dataset.

Table 38 shows four (4) samples of generated answers by SEQ2SEQ++ and benchmark models (STL, MTL-LTS, and MTL-BC). SEQ2SEQ++ model can reduce overall model overfitting by not generating frequently occurring words as part of the answers as the other models did and eventually improves the answer generation quality.

For example, in sample 1, incorrect answers were generated using benchmark models. STL generated the word "tree" which has a higher frequency of 99 instead of the word "guard." MTL-BC performed slightly better than STL because it can generate the word "Guard" correctly but incorrectly generated the end token "⟨end⟩" too early.

TABLE 38. Sample output SEQ2SEQ++ versus benchmark models.

Sample 1	**From NarrativeQA**	**BLEU Score**	**Frequency of term/phrase**
Question	what are marcus and his friends doing when the explosions go off ?		
Gold answer	a ***guard accidentally*** releases him from the cargo hold		"guard" : 51, "accidentally": 39
STL	a ***tree*** on fire	0.35	"tree":99
MTL-BC	a guard ***<end>***	0.49	"<end>" : 24000
MTL-LTS	a ***death*** deposit box	0.11	"death":232
SEQ2SEQ++	a guard accidentally releases him from the cargo hold	1.0	
Sample 2	**From NarrativeQA**	**BLEU Score**	**Frequency of term/phrase**
Question	where did flora live ?		
Gold answer	***summer country house*** in essex		"summer":22, "country":42 "house":263
STL	***into the*** country house in london	0.42	"into":302, "the":11902
MTL-BC	***in a*** bathtub in essex	0.36	"in":11991, "a":58556
MTL-LTS	summer country , minnesota	0.37	",":2040
SEQ2SEQ++	summer country house in essex	1.0	
Sample 3	**From SQuAD**	**BLEU Score**	**Frequency of term/phrase**
Question	What was Avicenna known for?		
Gold answer	His synthesis of knowledge from both ***east*** and west		"east":273
STL	***The*** kingdom of nations	0.11	"the":7479
MTL-BC	His synthesis of knowledge from both ***<end>***	0.61	"<end>": 24819
MTL-LTS	***The*** question brothers	0	"the":7479
SEQ2SEQ++	His synthesis of knowledge from both east and west	1.0	
Sample 4	**From SQuAD**	**BLEU Score**	**Frequency of term/phrase**
Question	What does Nanjing mean ?		
Gold answer	***Southern*** capital		"southern":65, "capital":33
STL	***The*** Great Britain	0	"the":7479
MTL-BC	Southern ***Greek***	0.58	"greek":77
MTL-LTS	***North*** America	0	"north":241
SEQ2SEQ++	Southern capital	1	

*General Note: All generated sentences will have <end> which symbolizes end of sentence, but it will not be shown to the user.

The end token has a frequency of 24,000 for NarrativeQA. MTL-LTS generated the wrong word ''death'' instead of the word ''guard.'' Similar outcomes were observed in the other samples.

This experiment shows that, while the other models generated frequently occurring words or tokens such as ''⟨end⟩,'' the comma, '','' ''a,'' ''in,'' and ''the,'' SEQ2SEQ++ avoided them to generate the correct words/tokens according to the respective questions.

V. CONCLUSION AND FUTURE WORK

Question-answering chatbots that provide concise answers to specific user questions and queries are rapidly gaining popularity in many domains, such as customer support and education. Neural network-based chatbot models equipped with domain knowledge can scale much faster than humans and can be utilized around the clock. It can be continuously trained with additional new data to be updated with the latest knowledge to be served to users.

The Seq2Seq natural answer generation method is one of the most popular methods for implementing question-answering chatbots. In this method, a question or answer is treated as a sequence of words or tokens. During training, the model learns to generate a sequence of words as answers given the question, which is also a sequence of words. Although the Seq2Seq based chatbots can provide answers to most questions, they tend to generate frequently occurring words in the answer; hence, the generated answer may not be relevant to the question. Some generated answers also ended abruptly, which means that it is not a complete answer. Consequently, the generated answers may be meaningless or unsatisfactory for the user. This Seq2Seq method's weakness can be attributed to three key issues: language model influence, answer generation overfit, and question encoding overfit.

The existing methods exhibit some gaps. First, existing attention methods only use the final hidden state of the decoder for decoding, and this may not be adequate to address the language model issue. Second, existing MTL models rely on fixed task loss weights for auxiliary tasks, which are not sufficient to reduce the answer generation overfitting. Third, the binary classification utilized by MTL-BC, which classifies an answer as positive or negative, is not natural. A more natural classification of the answer given a question is correct, partially correct, or incorrect. The MTL-LTS, which is the sequential MTL model, could not fully harness the power of the MTL as compared to the parallel MTL approach to reduce the question encoding overfit issue.

We propose four new methods to fill the gaps of Seq2Seq learning issues. CAM is a new attention mechanism that is utilized for decoding. The CAM considers all previous hidden states of the decoder during decoding. CAM can balance the gaps between language model influence and question influence. By creating this balance, the occurrence of high-frequency words can be reduced, and thus the model can generate a correct and meaningful answer. MFE performs first- and last-word prediction tasks in parallel with the answer-generation task. MFE, which is based on a parallel learning approach, has shown significant improvement over the sequential approach. Similarly, by utilizing TC, which performs ternary classification of the answer given a question, the question encoder overfit can also be reduced. Models that utilize MFE and TC can improve the answer generation quality by reducing the generation of high-frequency words incorrectly. CAM, MFE, and TC are integrated into a new MTL model called SEQ2SEQ++, which utilizes the DL weight mechanism, which calculates the task loss weights for each of the tasks in the MTL framework during the epoch and automatically uses it for the next epoch. This ensures that each task contributes accordingly to the overall model learning and ensures that model learning does not end prematurely.

We developed eight models and trained and tested each of them on two published research datasets to gauge the effectiveness of our proposed methods (CAM, DL, MFE, TC) and also our final model SEQ2SEQ++, which combines all our proposed methods to generate meaningful and relevant answers.

The results showed that our proposed methods (CAM, DL, MFE, TC) achieved better results than the benchmark models. The final experiment result showed that SEQ2SEQ++ achieved the best performance compared to the benchmark models in natural answer generation. SEQ2SEQ++ can produce answers with higher correctness (highest BLEU scores), lower errors (lowest WER scores), and higher diversity (highest Distinct-2 scores) on both datasets.

The significance of this research is three-fold:-

i) First, a comprehensive attention mechanism is proposed. CAM is generic and does not require additional input, so it can be used by researchers in other Seq2Seq-based tasks such as caption generation or question generation.

ii) Second, a DL weights scheme and a new DL-based MTL training algorithm. In an MTL model, determining the weight for each task's weight loss is not easy. Multiple arbitrary values must be assigned and tested before the final weight for a task can be identified. This trial-and-error approach is time-consuming and inefficient. It becomes even worse or nearly impossible if there are more than two tasks. Utilizing a dynamic task loss weight scheme is not only efficient but also highly effective in producing a better learning approach, as proved in this study. Moreover, this finding can encourage other researchers to further improve existing MTL models with more than one (1) auxiliary task. The DL-based MTL training algorithm can be readily adapted to any other parallel MTL framework.

iii) Third, this study confirms how additional tasks such as answer classification, first-word and last-word prediction tasks can be combined in a parallel MTL setting to improve the performance for the Seq2Seq learning-based answer generation. MFE shows how the question encoding overfit issue can be directly

addressed by performing further tasks on the just-on-the-question encoding alone without any need for additional data. TC shows how the answer can be classified in a more natural manner, which is effective in reducing question encoding overfit and is also valuable for natural language generation tasks. SEQ2SEQ++ is both a model and framework. As a model, researchers can utilize SEQ2SEQ++ to train their question–answer system in another domain or dataset. They can also perform their Seq2Seq-based NLP-based research in other areas such as question generation and translation. SEQ2SEQ++ is also a flexible framework. The existing auxiliary tasks (question–answer classification, first-word prediction, and last word prediction) can be replaced with other tasks, if needed. CAM can also be replaced with another attention mechanism that may be newly developed. This work also provided all the algorithms and formulas utilized for all the models implemented in this study. Researchers can replicate and implement them for benchmarking and further investigations.

In the future, we may devote our efforts to investigating SEQ2SEQ++ for multiturn conversations and other natural language tasks, such as question generation and summarization. We may even investigate how to further improve our model to show a much more significant improvement in diversity compared to other models. We are also interested in exploring pretrained language models such as the generative pretrained language models and embeddings such as BERT [23], GPT-3 [10], Word2Vec [57], and Glove [58] to be integrated into SEQ2SEQ++.

ACKNOWLEDGMENT

The authors would like to thank AFOSR for the support and express our appreciation to all who contributed to this research.

REFERENCES

[1] K. Palasundram, N. M. Sharef, K. A. Kasmiran, and A. Azman, "Enhancements to the sequence-to-sequence-based natural answer generation models," *IEEE Access*, vol. 8, pp. 45738–45752, 2020, doi: 10.1109/ACCESS.2020.2978551.

[2] M. Hardalov, I. Koychev, and P. Nakov, "Towards automated customer support," in *Proc. Int. Conf. Artif. Intell. Methodol. Syst. Appl.*, 2018, pp. 48–59.

[3] V. Hristidis, "Chatbot technologies and challenges," in *Proc. 1st Int. Conf. Artif. Intell. Ind. (AI4I)*, Sep. 2018, p. 126, doi: 10.1109/ai4i.2018.8665692.

[4] H.-Y. Shum, X.-D. He, and D. Li, "From Eliza to XiaoIce: Challenges and opportunities with social chatbots," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 1, pp. 10–26, Jan. 2018, doi: 10.1631/fitee.1700826.

[5] M. S. I. Bhuiyan, A. Razzak, M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque, and S. Tarkoma, "BONIK: A blockchain empowered chatbot for financial transactions," in *Proc. IEEE 19th Int. Conf. Trust. Secur. Privacy Comput. Commun. Trust.*, Dec. 2020, pp. 1079–1088, doi: 10.1109/TrustCom50675.2020.00143.

[6] S. Arsovski, S. Hui, and A. David, "Open-domain neural conversational agents: The step towards artificial general intelligence," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 6, pp. 403–408, 2018, doi: 10.14569/IJACSA.2018.090654.

[7] E. Merdivan, D. Singh, S. Hanke, and A. Holzinger, "Dialogue systems for intelligent human computer interactions," *Electron. Notes Theor. Comput. Sci.*, vol. 343, pp. 57–71, May 2019, doi: 10.1016/j.entcs.2019.04.010.

[8] G. Shubhashri, N. Unnamalai, and G. Kamalika, "LAWBO: A smart lawyer chatbot," in *Proc. ACM India Joint Int. Conf. Data Sci. Manage. Data*, 2018, pp. 348–351, doi: 10.1145/3152494.3167988.

[9] J. Walker, "Chatbot comparison—Facebook, Microsoft, Amazon, and Google," in *Techemergence*. 2018.

[10] T. Brown, B. Mann, N. Ryder, and M. Subbiah, "Language models are few-shot learners," 2020, *arXiv:2005.14165*.

[11] P. I. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, and L. Jones, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008, doi: 10.1109/2943.974352.

[12] A. Andrenucci and E. Sneiders, "Automated question answering: Review of the main approaches," in *Proc. 3rd Int. Conf. Inf. Technol. Appl. (ICITA)*, 2005, pp. 514–519, doi: 10.1109/ICITA.2005.78.

[13] T. Shao, Y. Guo, H. Chen, and Z. Hao, "Transformer-based neural network for answer selection in question answering," *IEEE Access*, vol. 7, pp. 26146–26156, 2019, doi: 10.1109/ACCESS.2019.2900753.

[14] S. Shang, J. Liu, and Y. Yang, "Multi-layer transformer aggregation encoder for answer generation," *IEEE Access*, vol. 8, pp. 90410–90419, 2020, doi: 10.1109/ACCESS.2020.2993875.

[15] T. Zhao, X. Lu, and K. Lee, "SPARTA: Efficient open-domain question answering via sparse transformer matching retrieval," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2021, pp. 565–575, doi: 10.18653/v1/2021.naacl-main.47.

[16] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 553–562, doi: 10.1145/2806416.2806493.

[17] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 3776–3783.

[18] Y. Huang and T. Zhong, "Multitask learning for neural generative question answering," *Mach. Vis. Appl.*, vol. 29, no. 6, pp. 1009–1017, Aug. 2018, doi: 10.1007/s00138-018-0908-0.

[19] Y. Wang, W. Rong, Y. Ouyang, and Z. Xiong, "Augmenting dialogue response generation with unstructured textual knowledge," *IEEE Access*, vol. 7, pp. 34954–34963, 2019, doi: 10.1109/ACCESS.2019.2904603.

[20] F. Liu, Q. Mao, L. Wang, N. Ruwa, J. Gou, and Y. Zhan, "An emotion-based responding model for natural language conversation," *World Wide Web*, vol. 22, no. 2, pp. 843–861, Mar. 2019, doi: 10.1007/s11280-018-0601-2.

[21] Y. Peng, Y. Fang, Z. Xie, and G. Zhou, "Topic-enhanced emotional conversation generation with attention mechanism," *Knowl.-Based Syst.*, vol. 163, pp. 429–437, Jan. 2019, doi: 10.1016/j.knosys.2018.09.006.

[22] M. Yang, W. Tu, Q. Qu, Z. Zhao, X. Chen, and J. Zhu, "Personalized response generation by dual-learning based domain adaptation," *Neural Netw.*, vol. 103, pp. 72–82, Jul. 2018, doi: 10.1016/j.neunet.2018.03.009.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.

[24] A. Katharopoulos, A. Vyas, and N. Pappas, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 5156–5165. [Online]. Available: http://proceedings.mlr.press/v119/katharopoulos20a.html

[25] M. Hahn, "Theoretical limitations of self-attention in neural sequence models," *Trans. Assoc. Comput. Linguist.*, vol. 8, pp. 156–171, 2020, doi: 10.1162/tacl_a_00306.

[26] R. Dale, "GPT-3: What's it good for?" *Natural Lang. Eng.*, vol. 27, no. 1, pp. 113–118, Jan. 2021, doi: 10.1017/S1351324920000601.

[27] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[28] M. Wei and Y. Zhang, "Natural answer generation with attention over instances," *IEEE Access*, vol. 7, pp. 61008–61017, 2019, doi: 10.1109/ACCESS.2019.2904337.

[29] M. T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–10.

[30] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, pp. 41–75, Dec. 1997, doi: 10.1007/978-3-030-01620-3_5.

[31] D. Dong, H. Wu, W. He, D. Yu, and H. Wang, "Multi-task learning for multiple language translation," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics*, 2015, pp. 1723–1732, doi: 10.3115/v1/p15-1166.

[32] R. Akama, K. Inada, N. Inoue, S. Kobayashi, and K. Inui, "Generating stylistically consistent dialog responses with transfer learning," in *Proc. 8th Int. Jt. Conf. Natural Lang. Process.*, vol. 2, 2017, pp. 408–412.
[33] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI*, 2017, pp. 2852–2858.
[34] M. Ghazvininejad, C. Brockett, and M. W. Chang, "A knowledge-grounded neural conversation model," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5110–5117.
[35] Z. Wang, Z. Wang, Y. Long, J. Wang, Z. Xu, and B. Wang, "Enhancing generative conversational service agents with dialog history and external knowledge," *Comput. Speech Lang.*, vol. 54, pp. 71–85, Mar. 2019, doi: 10.1016/j.csl.2018.09.003.
[36] D. Ren, Y. Cai, X. Lei, J. Xu, Q. Li, and H.-F. Leung, "A multi-encoder neural conversation model," *Neurocomputing*, vol. 358, pp. 344–354, Sep. 2019, doi: 10.1016/j.neucom.2019.05.071.
[37] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, D. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734, doi: 10.3115/v1/d14-1179.
[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, Sep. 2014, pp. 3104–3112.
[39] T. Koáiský, J. Schwarz, P. Blunsom, C. Dyer, K. Moritz Hermann, G. Melis, and E. Grefenstette, "The narrative QA reading comprehension challenge," 2017, *arXiv:1712.07040*.
[40] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuad: 100,000+questions for machine comprehension of text," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2016, pp. 2383–2392, doi: 10.18653/v1/d16-1264.
[41] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[42] Q. Zhu, W. Zhang, L. Zhou, and T. Liu, "Learning to start for sequence to sequence architecture," 2016, *arXiv:1608.05554*.
[43] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421, doi: 10.18653/v1/d15-1166.
[44] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," in *Proc. 7th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2015, pp. 1577–1586, doi: 10.3115/v1/p15-1152.
[45] C. Tao, S. Gao, M. Shang, W. Wu, D. Zhao, and R. Yan, "Get the point of my utterance! Learning towards effective responses with multi-head attention mechanism," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 4418–4424, doi: 10.24963/ijcai.2018/614.
[46] N. Asghar, P. Poupart, X. Jiang, and H. Li, "Deep active learning for dialogue generation," in *Proc. 6th Conf. Comput. Semant. Process.*, 2017, pp. 78–83, doi: 10.18653/v1/s17-1008.
[47] J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan, "A persona-based neural conversation model," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, vol. 2, 2016, pp. 994–1003, doi: 10.18653/v1/p16-1094.
[48] N. Asghar, P. Poupart, J. Hoey, X. Jiang, and L. Mou, "Affective neural response generation," in *Proc. Eur. Conf. Inf. Retr.*, vol. 10772, Sep. 2018, pp. 154–166, doi: 10.1007/978-3-319-76941-7_12.
[49] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meet. Assoc. Comput. Linguist.*, Jul. 2002, pp. 311–318, doi: 10.3115/1073083.1073135.
[50] Y. Zhang, M. Galley, J. Gao, Z. Gan, and X. Li, "Generating informative and diverse conversational responses via adversarial information maximization," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018, pp. 1–11.
[51] J. Xu, X. Ren, J. Lin, and X. Sun, "Diversity-promoting GaN: A cross-entropy based generative adversarial network for diversified text generation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3940–3949, 2020, doi: 10.18653/v1/d18-1428.
[52] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, Sep. 2010, pp. 1045–1048.
[53] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, Mar. 2016, pp. 110–119, doi: 10.18653/v1/n16-1014.
[54] M. Abadi, P. Barham, J. Chen, Z. Chen, and A. Davis, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement.*, 2016, pp. 265–268, doi: 10.1016/0076-6879(83)01039-3.
[55] E. Bisong, "Google colaboratory BT—Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners," Tech. Rep., 2019.
[56] A. K. Vijayakumar, M. Cogswell, and R. R. Selvaraju, "Diverse beam search for improved description of complex scenes," in *Proc. 32nd AAAI Conf. Artif. Intell. AAAI*, 2018, pp. 7371–7379.
[57] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed-representations-of-words-and-phrases-and-their-compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119, doi: 10.1162/jmlr.2003.3.4-5.951.
[58] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

KULOTHUNKAN PALASUNDRAM graduated from Universiti Kebangsaan Malaysia. He received the B.S. degree in computer science (Hons.) and the master's degree in IT, in 1995 and 1998, respectively. He is currently pursuing the Ph.D. degree in intelligent computing with Universiti Putra Malaysia. His research interests include artificial intelligence, deep learning, big data, natural language processing, and dialog generation.

NURFADHLINA MOHD SHAREF is currently an Associate Professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia. Her research interests include text mining, recommendation systems, and data science. Besides chatbot, her current projects are multi-objective deep reinforcement learning, multi-task deep learning for multi-class tweets classification, and deep-tensor factorization model for recommendation systems.

KHAIRUL AZHAR KASMIRAN received the Ph.D. degree from The University of Sydney, Australia, in 2012. He is currently a Senior Lecturer at the Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia. His interests include deep learning, reinforcement learning, performance engineering, formal verification, and software development.

AZREEN AZMAN (Member, IEEE) received the Diploma degree in software engineering from the Institute of Telecommunication and Information Technology, in 1997, the Bachelor of Information Technology degree in information systems engineering from Multimedia University, Malaysia, in 1999, and the Ph.D. degree in computing science specializing in information retrieval from the University of Glasgow, Scotland, in September 2007. Before joining his Ph.D. degree, he served in the industry for a few years. He is currently an Associate Professor at Universiti Putra Malaysia. His current research interests include information retrieval, text mining, natural language processing, and intelligent systems. He serves as a Committee Member for the Malaysian Society of Information Retrieval and Knowledge Management (PECAMP) and the Malaysian Information Technology Society (MITS).