The Context-dependent Additive Recurrent Neural Net

Quan Hung Tran [1,2] **Tuan Manh Lai** [2] **Gholamreza Haffari** [1]
Ingrid Zukerman[1] **Trung Bui** [2] **Hung Bui** [3]
[1] Monash University, Clayton, Australia
[2] Adobe Research, San Jose, CA
[3] DeepMind, Mountain View, CA

Abstract

Contextual sequence mapping is one of the fundamental problems in Natural Language Processing. Instead of relying solely on the information presented in a text, the learning agents have access to a strong external signal given to assist the learning process. In this paper, we propose a novel family of Recurrent Neural Network unit: the *Context-dependent Additive Recurrent Neural Network* (*CARNN*) that is designed specifically to leverage this external signal. The experimental results on public datasets in the dialog problem (Babi dialog Task 6 and Frame), contextual language model (Switchboard and Penn Discourse Tree Bank) and question answering (TrecQA) show that our novel CARNN-based architectures outperform previous methods.

1 Introduction

Sequence mapping is one of the most prominent class of problems in Natural Language Processing (NLP). This is due to the fact that written language is sequential in nature. In English, a word is a sequence of characters, a sentence is a sequence of words, a paragraph is a sequence of sentences, and so on. However, understanding a piece of text may require far more than just extracting the information from that piece itself. If the piece of text is a paragraph of a document, the reader may have to consider it together with other paragraphs in the document and the topic of the document. To understand an utterance in a conversation, the utterance has to be put into the context of the conversation, which includes the goals of the participants and the dialog history. Hence the notion of context is an intrinsic component of language understanding.

Inspired by recent works in dialog systems (Seo et al., 2017; Liu and Perez, 2017), we formalize the contextual sequence mapping problem as a sequence mapping problem with a strong controlling contextual element that regulates the flow of information. The system has two sources of signals: (i) the main text input, for example, the history utterance sequence in dialog systems or the sequence of words in language modelling; and (ii) *the context signal*, e.g., the previous utterance in a dialog system, the discourse information in contextual language modelling or the question in question answering.

Our contribution in this work is two-fold. First, we propose a new family of recurrent unit, the *Context-dependent Additive Recurrent Neural Network* (*CARNN*), specifically constructed for contextual sequence mapping. Second, we design novel neural network architectures based on CARNN for dialog systems and contextual language modelling, and enhance the state of the art architecture (IWAN (Shen et al., 2017)) on question answering. Our novel building block, the CARNN, draws inspiration from the Recurrent Additive Network (Lee et al., 2017), which showed that most of the non-linearity in the successful Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is not necessary. In the same spirit, our CARNN unit minimizes the use of non-linearity in the model to facilitate the ease of gradient flow. We also seek to keep the number of parameters to a minimum to improve train-ability.

We experiment with our models on a broad range of problems: dialog systems, contextual language modelling and question answering. Our systems outperform previous methods on several public datasets, which include the Babi Task 6 (Bordes and Weston, 2017) and the Frame dataset (Asri et al., 2017) for dialog, the Switchboard (Jurafsky et al., 1997) and Penn Discourse Tree Bank (Miltsakaki et al., 2004) for contextual language modelling, and the TrecQA

dataset (Wang et al., 2007) for question answering. We propose a different architecture for each task, but all models share the basic building block, the CARNN.

2 Background and Notation

Notation. As our paper describes several architectures with vastly different setups and input types, we introduce the following notation to maintain consistency and improve readability. First, the m-th input to the recurrent unit will be denoted $\mathbf{e}_m$. In language modelling, $\mathbf{e}_m$ is the embedding of the m-th word; while in dialog, it is the embedding of the m-th utterance (which is a combination of the embedding of the words inside the utterance, $\mathbf{x}_1^m \dots \mathbf{x}_{M_m}^m$). All the gates are denoted by $\mathbf{g}$, all the hidden vectors (outputs of the RNN) are denoted by $\mathbf{h}$. $\mathbf{W}s$ and $\mathbf{b}s$ are the RNN's parameters, σ denotes the sigmoid activation function, and $\odot$ denotes the element-wise product.

LSTM. The Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is arguably one of the most popular building blocks for RNN. The main components of the LSTM are three gates: an input gate $\mathbf{g}_m^i$ to regulate the information flow from the input to the memory cell $\mathbf{c}_m$, a forget gate $\mathbf{g}_m^f$ to regulate the information flow from the previous time step's memory cell $\mathbf{c}_{m-1}$, and an output gate $\mathbf{g}_m^o$ that regulates how the model produces the outputs (hidden state $\mathbf{h}_m$) from the memory cell $\mathbf{c}_t$. The computations of LSTM are as follows:

$$\begin{aligned}
\tilde{\mathbf{c}}_m &= tanh(\mathbf{W}_h^c \mathbf{h}_{m-1} + \mathbf{W}_x^c \mathbf{e}_m + \mathbf{b}_c) \\
\mathbf{g}_m^i &= \sigma(\mathbf{W}_h^i \mathbf{h}_{m-1} + \mathbf{W}_x^i \mathbf{e}_m + \mathbf{b}_i) \\
\mathbf{g}_m^f &= \sigma(\mathbf{W}_h^f \mathbf{h}_{m-1} + \mathbf{W}_x^f \mathbf{e}_m + \mathbf{b}_f) \\
\mathbf{g}_m^o &= \sigma(\mathbf{W}_h^o \mathbf{h}_{m-1} + \mathbf{W}_x^o \mathbf{e}_m + \mathbf{b}_o) \\
\mathbf{c}_m &= \mathbf{g}_m^i \odot \tilde{\mathbf{c}}_m + \mathbf{g}_m^f \odot \mathbf{c}_{m-1} \\
\mathbf{h}_m &= \mathbf{g}_m^o \odot tanh(\mathbf{c}_m)
\end{aligned} \tag{1}$$

RAN. The Recurrent Additive Neural Network (RAN) (Lee et al., 2017) is an improvement over the traditional LSTM. However, there are three major differences between the two. First, RAN simplifies the output computations by removing the output gate. Second, RAN simplifies the memory cell computations by removing the direct dependency between the candidate update memory cell $\tilde{\mathbf{c}}_m$ and the previous hidden vector $\mathbf{h}_{m-1}$. Finally, RAN removes the non-linearity from the transition dynamic of RNN by removing the $tanh$ non-linearity from the $\tilde{\mathbf{c}}_m$. The equations for RAN are as follows:

$$\begin{aligned}
\tilde{\mathbf{c}}_m &= \mathbf{W}_x^c \mathbf{e}_m \\
\mathbf{g}_m^i &= \sigma(\mathbf{W}_h^i \mathbf{h}_{m-1} + \mathbf{W}_x^i \mathbf{e}_m + \mathbf{b}_i) \\
\mathbf{g}_m^f &= \sigma(\mathbf{W}_h^f \mathbf{h}_{m-1} + \mathbf{W}_x^f \mathbf{e}_m + \mathbf{b}_f) \\
\mathbf{c}_m &= \mathbf{g}_m^i \odot \tilde{\mathbf{c}}_m + \mathbf{g}_m^f \odot \mathbf{c}_{m-1} \\
\mathbf{h}_m &= s(\mathbf{c}_m)
\end{aligned} \tag{2}$$

where s can be an identity function (identity RAN) or the $tanh$ activation function ($tanh$ RAN).

As shown in (Lee et al., 2017), RAN's memory cells $\mathbf{c}_m$ can be decomposed into a weighted sum of the inputs. Their experimental results show that RAN performs as well as LSTM for language modelling, while having significantly fewer parameters.

3 The Context-dependent Additive Recurrent Neural Net (CARNN)

In this section, we describe our novel recurrent units for the context-dependent sequence mapping problem.

Our RNN units use a different gate arrangement than that used by RAN. However, if we consider a broader definition of identity RAN, i.e., an RNN where hidden unit outputs can be decomposed into a weighted sum of inputs, where the weights are functions of the gates, then our first CARNN unit (nCARNN) can be viewed as an extension of identity RAN with additional controlling context.

The next two CARNN units (iCARNN and sCARNN) further simplify the nCARNN unit to improve train-ability.

3.1 Non-independent gate CARNN (nCARNN)

The main components of our recurrent units are the two gates (an update gate $\mathbf{g}^u$ and a reset gate $\mathbf{g}^f$), which jointly regulate the information from the input. The input vector, after being pushed through an affine transformation, is added into the previous hidden vector $\mathbf{h}_{m-1}$. The computations of the unit are as follows:

$$\begin{aligned}
\mathbf{g}_m^u &= \sigma(\mathbf{W}_u^c \mathbf{c} + \mathbf{W}_u^h \mathbf{h}_{m-1} + \mathbf{W}_u^e \mathbf{e}_m + \mathbf{b}_u) \\
\mathbf{g}_m^f &= \sigma(\mathbf{W}_f^c \mathbf{c} + \mathbf{W}_f^h \mathbf{h}_{m-1} + \mathbf{W}_f^e \mathbf{e}_m + \mathbf{b}_f) \\
\bar{\mathbf{e}}_m &= \mathbf{W}_{\bar{e}} \mathbf{e}_m + \mathbf{b}_{\bar{e}} \\
\mathbf{h}_m &= \mathbf{g}_m^u \odot (\mathbf{g}_m^f \odot \bar{\mathbf{e}}_m) + (1 - \mathbf{g}_m^u) \odot \mathbf{h}_{m-1}
\end{aligned} \tag{3}$$

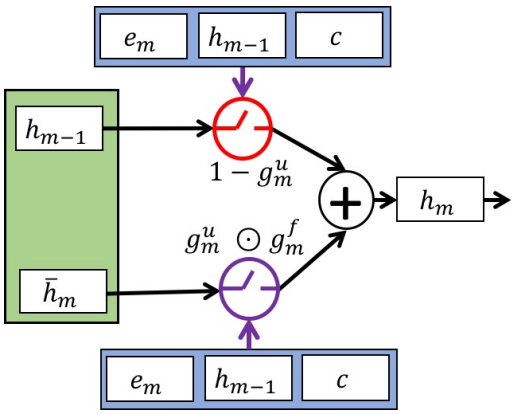

Figure 1: Context Dependent Additive Recurrent Neural Network. Note that only nCARNN has the previous hidden state $\mathbf{h}_{m-1}$ in its gate computation, iCARNN and sCARNN do not.

where $\mathbf{c}$ is the representation of the global context.

Apart from the non-linearity in the gates, our model is a linear function of the inputs. Hence, the final hidden layer of our RNN, denoted as $\mathbf{h}_M$, is a weighted sum of the inputs and a bias term $\mathbf{B}_i$ (Equation 4), where the weights are functions of the gates and $\mathbf{W}_{\bar{e}}$ is a dimension reduction matrix.

$$\begin{aligned}\mathbf{h}_M &= \mathbf{g}^u_M \odot \mathbf{g}^f_M \odot \bar{\mathbf{e}}_M + (1 - \mathbf{g}^u_M) \odot \mathbf{h}_{M-1} \\ &= \sum_{i=1}^{M} (\mathbf{g}^u_i \odot \mathbf{g}^f_i \odot \prod_{j=i+1}^{M} (1 - \mathbf{g}^u_j)) \odot \bar{\mathbf{e}}_i \\ &= \sum_{i=1}^{M} [(\mathbf{g}^u_i \odot \mathbf{g}^f_i \odot \prod_{j=i+1}^{M} (1 - \mathbf{g}^u_j)) \odot \mathbf{W}_{\bar{e}} \mathbf{e}_i + \mathbf{B}_i]\end{aligned} \tag{4}$$

From the decomposition in Equation 4, it seems that the outputs of an RNN with the nCARNN unit can be efficiently computed in parallel. That is, we can compute the weight for each input in parallel, and take their weighted sum to produce any desired hidden vector output. However, there is one obstacle: since the gates are functions of the previous hidden states, they still need to be computed sequentially. But if we assume that the external controlling context $\mathbf{c}$ is strong enough to regulate the flow of information, we can remove the previous hidden state (local context $\mathbf{h}_{m-1}$) from the gate computations, and make the RNN computations parallel. The next two variants of CARNN implement this idea by removing the local context from gate computations.

3.2 Independent gate CARNN (iCARNN)

The Gated Recurrent Unit (GRU) (Chung et al., 2014) and LSTM networks use a local context (the previous hidden state $\mathbf{h}_{m-1}$) and the current input to regulate the flow of information. In contrast, our model, relies on the global controlling context $\mathbf{c}$ at every step, and thus, might not need the local context $\mathbf{h}_{m-1}$ at all. Removing the local context can reduce the computational complexity of the model, but it may result in a loss of local sequential information. To test the effectiveness of this trade-off, we propose another variant of our unit, the *independent gate CARNN* (*iCARNN*), where the gate computations are simplified, and the gates are functions of the controlling context and the inputs. This formulation of CARNN is formally defined as follows.

$$\begin{aligned}\mathbf{g}^u_m &= \sigma(\mathbf{W}^c_u \mathbf{c} + \mathbf{W}^e_u \mathbf{e}_m + \mathbf{b}_u) \\ \mathbf{g}^f_m &= \sigma(\mathbf{W}^c_f \mathbf{c} + \mathbf{W}^e_f \mathbf{e}_m + \mathbf{b}_f) \\ \bar{\mathbf{e}}_m &= \mathbf{W}_{\bar{e}} \mathbf{e}_m + \mathbf{b}_{\bar{e}} \\ \mathbf{h}_m &= \mathbf{g}^u_m \odot (\mathbf{g}^f_m \odot \bar{\mathbf{e}}_m) + (1 - \mathbf{g}^u_m) \odot \mathbf{h}_{m-1}\end{aligned} \tag{5}$$

Compared to the traditional RNN, iCARNN's gates computations do not take into account the sequence context, i.e., the previous hidden vector computations, and the gates at all time steps can be computed in parallel. However, iCARNN, unlike memory network models (Sukhbaatar et al., 2015; Liu and Perez, 2017), still retains the sequential nature of RNN. This is because even though the gates at different time steps do not depend on each other, the hidden vector output at the m-th time step $\mathbf{h}_m$ depends on the previous gate ($\mathbf{g}^u_{m-1}$), and hence on the previous input.

3.3 Simplified candidate CARNN (sCARNN)

The standard GRU and the LSTM employ a linear transformation on the input representation before it is incorporated into the hidden representation. We have followed this convention with the previous variants of our unit. Although this transformation improves dimensional flexibility of the input/output vector, and adds representational power to the model with additional parameters, it also increases computational complexity. Fixing the output dimension to be the same as the input dimension makes it possible to reduce the computational complexity of the model. This leads us to propose another variant of the CARNN where the candidate update $\bar{\mathbf{e}}_m$ is the original embedding of the

current input (Equation 6). We call this variation the *simplified candidate CARNN* (*sCARNN*). The combination of lower gate computational complexity and the parallel-ability allow the paralleled sCARNN version to be 30% faster (30% lower training time for each epoch) than nCARNN in the question answering and dialog experiments, and 15% faster in the language model experiment. The sCARNN is formally defined as follows.

$$\begin{aligned}\mathbf{g}_m^u &= \sigma(\mathbf{W}_u^c\mathbf{c} + \mathbf{W}_u^e\mathbf{e}_m + \mathbf{b}_u)\\ \mathbf{g}_m^f &= \sigma(\mathbf{W}_f^c\mathbf{c} + \mathbf{W}_f^e\mathbf{e}_m + \mathbf{b}_f)\\ \mathbf{h}_m &= \mathbf{g}_m^u \odot (\mathbf{g}_m^f \odot \mathbf{e}_m) + (1 - \mathbf{g}_m^u) \odot \mathbf{h}_{m-1}\end{aligned} \tag{6}$$

sCARNN can still be decomposed into a weighted sum of the sequence of input elements, and retains the parallel computation capability of the iCARNN.

$$\begin{aligned}\mathbf{h}_M &= \mathbf{g}_M^u \odot \mathbf{g}_M^f \odot \mathbf{e}_m + (1 - \mathbf{g}_M^u) \odot \mathbf{h}_{M-1}\\ &= \sum_{i=1}^{M}(\mathbf{g}_i^u \odot \mathbf{g}_i^f \odot \prod_{j=i+1}^{M}(1 - \mathbf{g}_j^u)) \odot \mathbf{e}_i\end{aligned} \tag{7}$$

4 CARNN-based models for NLP problems

In this section, we explain the details of our CARNN-based architectures for end-to-end dialog, language modelling and question answering. In each of these applications, one of the main design concerns is the choice of contextual information. As we will demonstrate in this section, the controlling context $\mathbf{c}$ can be derived from various sources: a sequence of words (dialog and question answering), a class variable (language modelling). Virtually any sources of strong information that can be encoded into vectors can be used as controlling context.

4.1 End-to-end dialog

To produce a response, we first encode the whole dialog history into a real vector representation $\mathbf{h}_{his}$. To this effect, we perform two steps: first, we encode each utterance (sequence of words) into a real vector, and next, we encode this sequence of real vector representations into $\mathbf{h}_{his}$. We employ the Position Encoder (Bordes and Weston, 2017) for the first step, and CARNNs for the second step.

Summarizing individual utterances. Let's denote the sequence of word-embeddings in the m-th utterance $\mathbf{x}_1^m, \dots \mathbf{x}_{N_m}^m$. These word embeddings are jointly trained with the model. Following previous work in end-to-end dialog systems, we opt to use the Position Encoder (Liu and Perez, 2017; Bordes and Weston, 2017) for encoding utterances.

The Position Encoder is an improvement over the average embedding of bag of words, as it takes into account the position of the words in a sequence. This encoder has been empirically shown to perform well on the Babi dialog task (Liu and Perez, 2017; Bordes and Weston, 2017); more details about the Position Encoder can be found in (Sukhbaatar et al., 2015). Let's denote the the embeddings of a sequence of utterances $\mathbf{e}_1, \dots \mathbf{e}_{M-1}$.

Summarizing the dialog history. The CARNN models take the embeddings of the sequence of utterances and produce the final representation $\mathbf{h}_{his}$. We further enhance the output of the CARNN by adding the residual connection to the input (He et al., 2016; Tran et al., 2017), and the attention mechanism (Bahdanau et al., 2015) over the history.

$$\begin{aligned}\mathbf{h}_1, ..\mathbf{h}_{M-1} &= CARNN(\mathbf{e}_1, ..\mathbf{e}_{M-1}, \mathbf{c})\\ \forall m \in [1..M-1] : \tilde{\mathbf{h}}_m &= \mathbf{h}_m + \mathbf{e}_m\\ \alpha_1..\alpha_{M-1} &= softmax(\tilde{\mathbf{h}}_1^T\mathbf{c}, .., \tilde{\mathbf{h}}_{M-1}^T\mathbf{c})\\ \mathbf{h}_{his} &= \sum_{m=1}^{M-1}\alpha_m\tilde{\mathbf{h}_m}\end{aligned} \tag{8}$$

where α are the attention weights, $\mathbf{h}_m$ is the m-th output of the base CARNN, $\mathbf{e}_m$ is the embedding of the m-th input utterance, and $\mathbf{c} = \mathbf{e}_M$ is the context embedding.

Our model chooses the response from a set of pre-determined system answers (a task setup following Bordes and Weston (2017); Liu and Perez (2017); Seo et al. (2017)). However, in the dialog case, the answers themselves are sequences of words, and treating them as distinct classes may not be the best approach. In fact, previous work in memory networks (Liu and Perez, 2017; Bordes and Weston, 2017) employs a feature function Φ to extract features from the candidate responses. In our work, we do not use any feature extraction, and simply use the Position Encoder to encode the

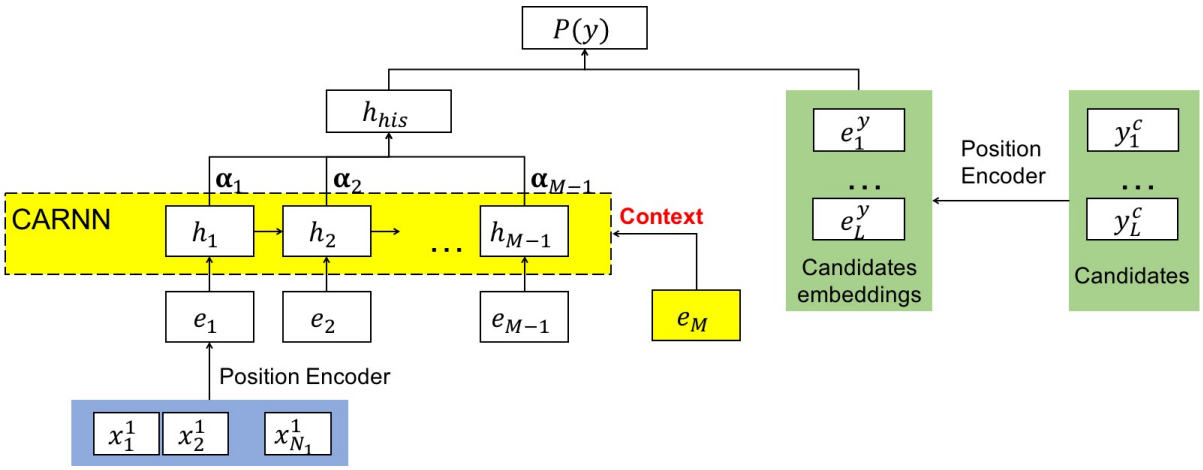

Figure 2: CARNN for dialog.

responses as shown in Figure 2, which depicts our architecture of CARNN for dialog.

$$\forall l \in [1..L] : \mathbf{e}_l = Position\_Encoder(y_l^c) \quad (9)$$

We then put a distribution over the candidate responses conditioned on the summarized dialog history $\mathbf{h}_{his}$ (Equation 10).

$$\mathbb{P}(y) = softmax(\mathbf{h}_{his}^T \mathbf{e}_1^y, ..., \mathbf{h}_{his}^T \mathbf{e}_L^y) \quad (10)$$

4.2 Contextual language model

Typically, language models operate at the sentence level, i.e., the sentences are treated independently. Several researchers have explored inter-sentence and inter-document level contextual information for language modelling (Ji et al., 2016a,b; Tran et al., 2016; Lau et al., 2017).

Following Ji et al. (2016a,b), we investigate two types of contextual information: (i) the previous sentence context; and (ii) a latent variable capturing the connection information between sentences, such as discourse relation in the Penn Discourse Tree Bank dataset or Dialog Acts in the Switchboard dataset.

Previous sentence context. The previous sentence (time-step $t-1$) contextual information is encoded by a simplified version of the nCARNN, where the global context is absent. The final hidden vector of this sequence is then fed into the current recurrent computation (time-step t) as the context for that sequence. Equation 11 shows this procedure.

$$\begin{aligned} \mathbf{c}^{t-1} &\longleftarrow nCARNN(\mathbf{e}_1^{t-1}, ..\mathbf{e}_{M^{t-1}}^{t-1}) \\ \mathbf{h}_1^t, ..\mathbf{h}_{M^{t-1}}^t &= CARNN(\mathbf{e}_1^t, ..\mathbf{e}_{M^t}^t, \mathbf{c}^{t-1}) \\ w_{m+1}^t &\sim softmax(\mathbf{W}^{(l)}\mathbf{h}_m^t + \mathbf{b}^{(l)}) \end{aligned} \quad (11)$$

Latent variable context. Ji et al. (2016b) proposed to embed the predicted latent variables using an embedding matrix, and use this real vector as the contextual information. In our work, we design a multi-task learning scenario where the previous sentence context encoder has additional supervised information obtained from the annotated latent variable (L^{t-1}). This additional information from the latent variable is only used to train the previous sentence encoder, and enhance the context $\mathbf{c}^{t-1}$ (Equation 12). During test time, the language model uses the same computation steps as the previous sentence context version.

$$\begin{aligned} \mathbb{P}(L^{t-1}) &= softmax(\mathbf{W}^{(c)}\mathbf{c}^{t-1} + \mathbf{b}^{(c)}) \\ \mathbb{L}_l^{t-1} &\sim \mathbb{P}(L^{t-1}) \end{aligned} \quad (12)$$

During training, the total loss function ($\mathbb{L}_{l,w}^t$) is the linear combination of the average log-loss from the current sentence's words ($\mathbb{L}_w^t$) and the log-loss from the previous latent variable ($\mathbb{L}_l^{t-1}$).

$$\mathbb{L}_{l,w}^t = \alpha \mathbb{L}_w^t + (1-\alpha)\mathbb{L}_l^{t-1} \quad (13)$$

where α is a linear mixing parameter. In our experiments, tuning α does not yield significant improvements, hence we set $\alpha = 0.5$.

4.3 Question answering

Answer selection is an important component of a typical question answering system. This task can be briefly described as follows: Given a question q and a candidate set of sentences $c_1, c_2, \ldots c_n$, the goal is to identify positive sentences that contain the answer. Many researchers have investigated employing neural networks for this task (Rao

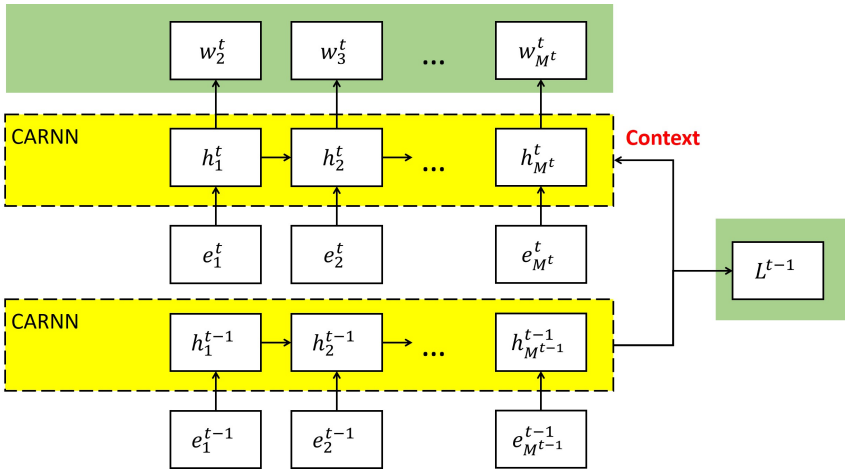

Figure 3: CARNN for context-dependent language model.

et al., 2016; Wang et al., 2017; Bian et al., 2017; Shen et al., 2017; Tay et al., 2017; He et al., 2015). Below is an example from the answer selection TrecQA corpus:

Question: Who established the Nobel prize awards?
Positive answer: The Nobel Prize was established in the will of Alfred Nobel, a Swede who invented dynamite and died in 1896.
Negative answer: The awards aren't given in specific categories.

The IWAN model proposed in (Shen et al., 2017) achieves state-of-the-art performance on the Clean version TrecQA dataset (Wang et al., 2007) for answer selection. In general, given two sentences, the model aims to calculate a score to measure their similarity. For each sentence, the model first uses a bidirectional LSTM to obtain a context-aware representation for each position in the sentence. The representations will later be utilized by the model to compute similarity score of the two sentences according to the degree of their alignment (Shen et al., 2017).

The original IWAN model employed LSTM to encode the sentence pair into sequences of real vector representations. However, these sequences are independent, and do not take into account the information from the other sentence. In order to overcome this limitation, we enhance the IWAN model with a "cross context CARNN-based sentence encoder" that replaces the bidirectional LSTM. When the cross context CARNN sentence encoder processes a sentence, it takes the encoding of the other sentence, encoded by a Position Encoder, as the controlling context (Figure 4).

5 Experiments

5.1 End-to-end dialog

Datasets. For the dialog experiments, we focus on two popular datasets for dialog: the Babi dataset (Bordes and Weston, 2017) and the Malluba Frame dataset (Asri et al., 2017).[1]

In our main set of experiments for dialog, we use the original Babi task 6 dataset, and test on the end-to-end dialog setting (the same setting used by Seo et al. (2017); Bordes and Weston (2017); Liu and Perez (2017)). That is, the systems have to produce complete responses and learn the dialog behaviour solely from the ground truth responses without help from manual features, rules or templates. Apart from this main set of experiments, we apply our end-to-end systems as dialog managers and test on a slightly different setting in the next two sets of experiments.

In the second set of experiments, we use our end-to-end systems as "dialog managers". The only difference compared to the end-to-end dialog setting is that the systems produce templatized responses instead of complete responses. Our motivation for this dialog manager setting is that in our preliminary experiments with the Babi dataset, we found out that many of the classification errors are due to very closely related responses, all of which fit the corresponding context. We argue that if we treat the systems as dialog managers, then we can delexicalize and group similar responses. Thus following Williams et al. (2017), we construct a templatized set of responses. For example, all the

[1] Among the Babi tasks, we focus mainly on task 6, which is based on real human-machine interactions. The other five Babi datasets comprise synthetically generated data.

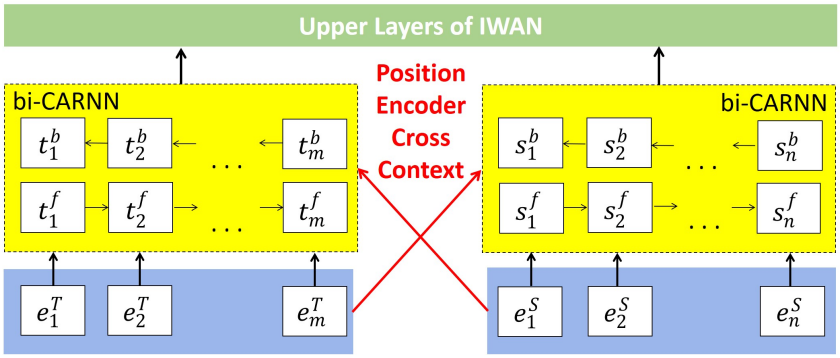

Figure 4: CARNN for Question Answering.

responses similar to *"**india_house** is in the **west** part of town"* will be grouped into *"**_name_** is in the **_loc_** part of town"*. The set of responses is reduced to 75 templatized responses. We call this new dataset "Babi reduced".[2]

The third set of experiments is conducted on the Frame dataset. The general theme in this dataset is similar to that of the Babi task 6, but the responses in the Frame dataset are generally in free form, rather than being sourced from a limited set. Thus, we define a dialog task on the Frame data set similar to the Babi reduced dialog task by simplifying and grouping the responses.[3] The final set of responses consists of 129 response classes. For the experiments on the Frame dataset, we randomly choose 80% of the conversations as the training set, and 10% each for testing and development.

Baselines. In the dialog experiments, we focus on the existing published results with end-to-end settings, namely the Memory Network (MN) (Bordes and Weston, 2017), the Gated Memory Network (GMN) (Liu and Perez, 2017) and the Query Reduction Network (QRN) (Seo et al., 2017).[4] For the Frame and Babi reduced datasets, we use the publicly available implementation of the QRN,[5] and our implementation of the GMN with hyper parameters similar to those reported by Liu and Perez (2017); Seo et al. (2017). Note that the original results presented by Seo et al. (2017), take into account partial matches (matching only a portion of the ground truth response), and hence cannot be directly translated into the standard response accuracy reported by other researchers (we have confirmed this with Seo *et al.*). For a direct comparison with the QRN, we use the evaluation settings employed in other papers (Liu and Perez, 2017; Sukhbaatar et al., 2015).

Model	Babi	Babi reduced	Frame
nCARNN	51.3%*	55.8%*	27.4%*
iCARNN	52.0%*	55.2%*	28.5%*
sCARNN	50.9%*	55.9%*	25.7%*
CARNN voting	**53.2**%*	**56.9**%*	**29.1**%*
QRN (2017)	46.8%	54.7%	24.0%
GMN (2017)	47.4%	54.1%	23.6%
MN (2017)	41.1%	–	–

Table 1: Dialog accuracy on Babi and Frame among end-to-end systems. * indicates statistical significance with $p < 0.1$ compared to QRN.

Results and discussion. Table 1 shows the results of the end-to-end models for the dialog task. All the CARNN-based systems are implemented in Tensorflow (Abadi et al., 2015) with a hidden vector size of 1024. As seen in Table 1, our models achieve the best results, and within the variants of our models, the iCARNN either performs the best, or very close to the best on all datasets. Majority voting provides a significant boost to the performance of the CARNN models. Upon comparison with the baseline systems, CARNN models tend to perform better on instances which require the system to remember specific information through a long dialog history. In Figure 5, the user already mentioned that he/she wants to find a "cheap" restaurant, but the GMN and QRN seem to "forget" this information. We speculate that due

[2] We do not have access to Williams et al. (2017)'s template set, thus the results in Babi reduced are not comparable to those obtained by Williams et al. (2017).

[3] We use only one of the annotated *"Dialog acts"* and its first slot key as a template for the response.

[4] Williams et al. (2017) and Liu and Lane (2017) reported very strong performances (55.6% and 52.8% respectively) for the Babi dataset. However, these systems do not learn dialog behaviour solely from Babi's ground truth responses, and thus do not have end-to-end dialog setups. As stated in their papers, Williams *et al.*use hand-coded rules and task-specific templates, while Liu *et al.*employ the external users' goal annotations that are outside the Babi dataset.

[5] https://github.com/uwnlp/qrn

U: im looking for a cheap restaurant
S: ... What type of food do you want??
...5 dialog turns...
S: Could you please repeat that?
U: vietnamese food

CARNN action: api_call vietnamese R_location cheap
QRN action: api_call vietnamese R_location R_price
GMN action: api_call vietnamese R_location R_price

Figure 5: Sample dialog from our system compared to the baselines. Only CARNN's predicted action takes into account the original cheap restaurant request and matches the ground truth action (in the systems' api calls, "R_price" denotes "any price").

to the ease of training, CARNN models summarize the dialog history better, and allow for longer information dependency.

The CARNN units are originally designed in the dialog context. During model calibration, we also tested in the dialog experiments two other CARNN versions with both higher and lower complexity. The lower complexity CARNN version resembles sCARNN without the forget gate, and the higher complexity CARNN version resembles the LSTM unit with all three gates (forget, update and output), with the gates being modified from the original LSTM gates to be functions of the external contextual information. Both of these versions do not perform as well as the three main CARNN versions (48.7% and 48.6% for the high- and low-complexity versions respectively in the Babi task).

5.2 Contextual language model

Datasets. We employ two datasets for the experiments with the contextual language model: the Switchboard Dialog Act corpus and the Penn Discourse Tree Bank corpus. There are 1155 telephone conversations in the Switchboard corpus, where each conversation has an average of 176 utterances. There were originally 226 Dialog Act (DA) labels in the corpus, but they are usually clustered into 42 labels. The Penn Tree Bank corpus provides discourse relation annotation between the spans of text. We used the preprocessed data by Ji et al. (2016b), where the explicit discourse relations are mapped into a dummy relation. Our data splits are the same as those described in the baselines (Ji et al., 2016a,b).

Baselines. We compare our system with the Recurrent Neural Net (RNNLM) with LSTM unit (Ji et al., 2016a), the Document Contextual Language Model (DCLM) (Ji et al., 2016a) and the Discourse Relation Language Model (DRLM) (Ji et al., 2016b). The RNNLM's architecture is the same as that described in (Mikolov et al., 2013) with sigmoid non-linearity replaced by LSTM. The DCLM exploits the inter-sentences context by concatenating the representation of the previous sentence with the input vector (context-to-context) or the hidden vector (context-to-output). The DRLM introduces the latent variable contextual models using a generative architecture that treats Dialog Acts or discourse relations as latent variables.

Model	Penn Discourse Tree Bank	Switchboard
nCARNN (w/o latent)	96.95	30.17
iCARNN (w/o latent)	94.72	32.49
sCARNN (w/o latent)	87.39	31.50
nCARNN (with latent)	96.64	**29.72**
iCARNN (with latent)	94.16	32.16
sCARNN (with latent)	**86.68**	31.49
RNNLM (2016b)	117.8	56.0
DCLM (2016a)	112.2	45.3
DRLM (2016b)	108.3	39.6

Table 2: Perplexity on Switchboard and Penn Discourse Tree Bank.

Results and discussion. Table 2 shows the test set perplexities across the systems for the Penn Tree Bank and Switchboard datasets. Interestingly, in these experiments, the system with the least computational complexity, the sCARNN, performs best on Penn Discourse Tree Bank, and second best on Switchboard. Generally, we found out that adding the Dialog Act/Discourse supervised signal in a multi-task learning scheme provides a boost to performance, but this improvement is small.

5.3 Question answering

Datasets. The TrecQA dataset (Wang et al., 2007) is a widely-used benchmark for answer selection. There are two versions of TrecQA: original and clean. The original TrecQA consists of 1,229 training questions, 82 development questions, and 100 test questions. Recently, researchers (Rao et al., 2016; Shen et al., 2017) developed a clean version, where they removed questions in the development and test sets with no answers or only positive/negative answers. This reduced the development and test set's sizes to 65

Model	MAP	MRR
IWAN (our implementation) + nCARNN*	0.827	0.889
IWAN (our implementation) + iCARNN*	0.826	**0.907**
IWAN (our implementation) + sCARNN*	**0.829**	0.875
IWAN (our implementation)	0.794	0.879
IWAN (2017)	0.822	0.889
Compare-Aggregate (2017)	0.821	0.899
BiMPM (2017)	0.802	0.875
NCE-CNN (2016)	0.801	0.877
HyperQA (2017)	0.784	0.865

Table 3: MAP and MRR for question answering. * indicates statistical significance with $\alpha < 0.05$ in t-test compared to IWAN (our implementation).

and 68 questions respectively.

Baselines. We compare the performance of our models with that of the state-of-the-art models on the clean version of the TREC-QA dataset (Shen et al., 2017; Bian et al., 2017; Wang et al., 2017; Rao et al., 2016; Tay et al., 2017). We do not have access to the original implementation of IWAN, hence we use our implementation of the IWAN model as the basis for our models.

Results and discussion. Table 3 shows the MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank) of our systems and the baselines. To the best of our knowledge, our systems outperform all previous systems on this dataset. Enhancing IWAN with cross-context CARNN statistically significantly improves performance. Among the variants, the iCARNN is the most consistent in both MAP and MRR. During our error analysis, we noted that the top answer returned by IWAN models with either LSTM or CARNNs are usually good. However, in many cases, lower ranked answers returned by the LSTM model are not as good as those produced by the CARNN models. We show an example of this in Table 4.

Question: **During what war did Nimitz serve?**

IWAN-LSTM	IWAN-iCARNN
Since the museum opened in 1983, Fredericksburg has become a haven for retired military servicemen who come to trace Nimitz's career and the events of World War II.	Since the museum opened in 1983, Fredericksburg has become a haven for retired military servicemen who come to trace Nimitz's career and the events of World War II.
Bill McCain, who graduated from West Point, chased Pancho Villa with Gen. Blackjack Pershing, served as an artillery officer during World War I and attained the rank of brigadier general.	Indeed, the ancestors of Chester W. Nimitz, the U.S. naval commander in chief of the Pacific in World War II, were among the first German pioneers to settle the area.
There was his grandfather, Admiral John "Slew" McCain, Class of 1906, a grizzled old sea dog who commanded aircraft carriers in the Pacific during World War II.	Slew McCain's peers at the Naval Academy were Chester Nimitz and William "Bull" Halsey, who would become major commanders during World War II.

Table 4: Top three answers produced by CARNN and LSTM. Blue colored answers are correct and red ones are incorrect.

6 Conclusion and future work

In this paper, we propose a novel family of RNN units which are particularly useful for the contextual sequence mapping problem: the CARNNs. Together with our neural net architectures, CARNN-based systems outperform previous methods on several public datasets for dialog (Frame and Babi Task 6), question answering (TrecQA) and contextual language modelling (Switchboard and Penn Discourse Tree Bank). In the future, we plan to investigate the effectiveness of CARNN units in other sequence modelling tasks.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection.

In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. pages 1987–1990.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *ICLR 2017*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*. pages 1576–1586.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

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

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016a. Document context language models. In *ICLR (Workshop track)*.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016b. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 332–342.

Daniel Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. Technical report, University of Colorado.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 355–365.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *arXiv preprint arXiv:1705.07393* .

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Interspeech 2017*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1–10.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2004. The penn discourse treebank. In *LREC*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, pages 1913–1916.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-regression networks for machine comprehension. In *ICLR*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1190–1200.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR* abs/1707.07847.

Quan Tran, Andrew MacKinlay, and Antonio Jimeno Yepes. 2017. Named entity recognition with stack residual lstm and trainable bias decoding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 566–575.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2016. Inter-document contextual language model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 762–766.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*. volume 7, pages 22–32.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814* .

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vancouver, Canada, pages 665–677.