Conditional Generation and Snapshot Learning in Neural Dialogue Systems

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Steve Young
Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK
{thw28,mg436,nm480,lmr46,phs26,su259,djv27,sjy11}@cam.ac.uk

Abstract

Recently a variety of LSTM-based conditional language models (LM) have been applied across a range of language generation tasks. In this work we study various model architectures and different ways to represent and aggregate the source information in an end-to-end neural dialogue system framework. A method called snapshot learning is also proposed to facilitate learning from supervised sequential signals by applying a *companion* cross-entropy objective function to the conditioning vector. The experimental and analytical results demonstrate firstly that competition occurs between the conditioning vector and the LM, and the differing architectures provide different trade-offs between the two. Secondly, the discriminative power and transparency of the conditioning vector is key to providing both model interpretability and better performance. Thirdly, snapshot learning leads to consistent performance improvements independent of which architecture is used.

1 Introduction

Recurrent Neural Network (RNN)-based conditional language models (LM) have been shown to be very effective in tackling a number of real world problems, such as machine translation (MT) (Cho et al., 2014) and image caption generation (Karpathy and Fei-Fei, 2015). Recently, RNNs were applied to task of generating sentences from an explicit semantic representation (Wen et al., 2015a). Attention-based methods (Mei et al., 2016) and Long Short-term Memory (LSTM)-like (Hochreiter and Schmidhuber, 1997) gating mechanisms (Wen et al., 2015b) have both been studied to improve generation quality. Although it is now clear that LSTM-based conditional LMs can generate plausible natural language, less effort has been put in comparing the different model architectures. Furthermore, conditional generation models are typically tested on relatively straightforward tasks conditioned on a single source (e.g. a sentence or an image) and where the goal is to optimise a single metric (e.g. BLEU). In this work, we study the use of conditional LSTMs in the generation component of neural network (NN)-based dialogue systems which depend on multiple conditioning sources and optimising multiple metrics.

Neural conversational agents (Vinyals and Le, 2015; Shang et al., 2015) are direct extensions of the sequence-to-sequence model (Sutskever et al., 2014) in which a conversation is cast as a source to target transduction problem. However, these models are still far from real world applications because they lack any capability for supporting domain specific tasks, for example, being able to interact with databases (Sukhbaatar et al., 2015; Yin et al., 2016) and aggregate useful information into their responses. Recent work by Wen et al. (2016a), however, proposed an end-to-end trainable neural dialogue system that can assist users to complete specific tasks. Their system used both distributed and symbolic representations to capture user intents, and these collectively condition a NN language generator to generate system responses. Due to the diversity of the conditioning information sources, the best way to represent and combine them is non-trivial.

In Wen et al. (2016a), the objective function for learning the dialogue policy and language generator depends solely on the likelihood of the output sentences. However, this sequential supervision signal may not be informative enough to learn a good conditioning vector representation resulting in a generation process which is dominated by the LM. This can often lead to inappropriate system outputs.

In this paper, we therefore also investigate the use of snapshot learning which attempts to mitigate this problem by heuristically applying *companion* supervision signals to a subset of the conditioning vector. This idea is similar to deeply supervised nets (Lee et al., 2015) in which the final cost from the output layer is optimised together with the companion signals generated from each intermediary layer. We have found that snapshot learning offers several benefits: (1) it consistently improves performance; (2) it learns discriminative and robust feature representations and alleviates the vanishing gradient problem; (3) it appears to learn transparent and interpretable subspaces of the conditioning vector.

2 Related Work

Machine learning approaches to task-oriented dialogue system design have cast the problem as a partially observable Markov Decision Process (POMDP) (Young et al., 2013) with the aim of using reinforcement learning (RL) to train dialogue policies online through interactions with real users (Gašić et al., 2013). In order to make RL tractable, the state and action space must be carefully designed (Young et al., 2010) and the understanding (Henderson et al., 2014; Mrkšić et al., 2015) and generation (Wen et al., 2015b; Wen et al., 2016b) modules were assumed available or trained standalone on supervised corpora. Due to the underlying hand-coded semantic representation (Traum, 1999), the conversation is far from natural and the comprehension capability is limited. This motivates the use of neural networks to model dialogues from end to end as a conditional generation problem.

Interest in generating natural language using NNs can be attributed to the success of RNN LMs for large vocabulary speech recognition (Mikolov et al., 2010; Mikolov et al., 2011). Sutskever et al. (2011) showed that plausible sentences can be obtained by sampling characters one by one from the output layer of an RNN. By conditioning an LSTM on a sequence of characters, Graves (2013) showed that machines can synthesise handwriting indistinguishable from that of a human. Later on, this idea has been tried in several research fields, for example, generating image captions by conditioning an RNN on a convolutional neural network (CNN) output (Karpathy and Fei-Fei, 2015; Xu et al., 2015); translating a source to a target language by conditioning a decoder LSTM on top of an encoder LSTM (Cho et al., 2014; Bahdanau et al., 2015); or generating natural language by conditioning on a symbolic semantic representation (Wen et al., 2015b; Mei et al., 2016). Among all these methods, attention-based mechanisms (Bahdanau et al., 2015; Hermann et al., 2015; Ling et al., 2016) have been shown to be very effective improving performance using a dynamic source aggregation strategy.

To model dialogue as conditional generation, a sequence-to-sequence learning (Sutskever et al., 2014) framework has been adopted. Vinyals and Le (2015) trained the same model on several conversation datasets and showed that the model can generate plausible conversations. However, Serban et al. (2015b) discovered that the majority of the generated responses are generic due to the maximum likelihood criterion, which was latter addressed by Li et al. (2016a) using a maximum mutual information decoding strategy. Furthermore, the lack of a consistent system persona was also studied in Li et al. (2016b). Despite its demonstrated potential, a major barrier for this line of research is data collection. Many works (Lowe et al., 2015; Serban et al., 2015a; Dodge et al., 2016) have investigated conversation datasets for developing chat bot or QA-like general purpose conversation agents. However, collecting data to develop goal oriented dialogue systems that can help users to complete a task in a specific domain remains difficult. In a recent work by Wen et al. (2016a), this problem was addressed by designing an online, parallel version of Wizard-of-Oz data collection (Kelley, 1984) which allows large scale and cheap in-domain conversation data to be collected using Amazon Mechanical Turk. An NN-based dialogue model was also proposed to learn from the collected dataset and was shown to be able to assist human subjects to complete specific tasks.

Snapshot learning can be viewed as a special form of weak supervision (also known as distant- or self supervision) (Craven and Kumlien, 1999; Snow et al., 2004), in which supervision signals are heuristically labelled by matching unlabelled corpora with entities or attributes in a structured database. It has been widely applied to relation extraction (Mintz et al., 2009) and information extraction (Hoffmann et al., 2011) in which facts from a knowledge base (e.g. Freebase) were used as objectives to train classifiers. Recently, self supervision was also used in memory networks (Hill et al., 2016) to improve the discriminative power of memory attention. Conceptually, snapshot learning is related to curriculum learning (Bengio et al., 2009). Instead of learning easier examples before difficult ones, snapshot learning creates an easier target for each example. In practice, snapshot learning is similar to deeply supervised nets (Lee et al., 2015) in which *companion* objectives are generated from intermediary layers and optimised altogether with the output objective.

3 Neural Dialogue System

The testbed for this work is a neural network-based task-oriented dialogue system proposed by Wen et al. (2016a). The model casts dialogue as a source to target sequence transduction problem (modelled by a sequence-to-sequence architecture (Sutskever et al., 2014)) augmented with the dialogue history (modelled by a belief tracker (Henderson et al., 2014)) and the current database search outcome (modelled by a database operator). The model consists of both encoder and decoder modules. The details of each module are given below.

3.1 Encoder Module

At each turn t, the goal of the encoder is to produce a distributed representation of the system action $\mathbf{m}_t$, which is then used to condition a decoder to generate the next system response in skeletal form[1]. It consists of four submodules: intent network, belief tracker, database operator, and policy network.

Intent Network The intent network takes a sequence of tokens[1] and converts it into a sentence embedding representing the user intent using an LSTM network. The hidden layer of the LSTM at the last encoding step $\mathbf{z}_t$ is taken as the representation. As mentioned in Wen et al. (2016a), this representation can be viewed as a distributed version of the speech act (Traum, 1999) used in traditional systems.

Belief Trackers In addition to the intent network, the neural dialogue system uses a set of slot-based belief trackers (Henderson et al., 2014; Mrkšić et al., 2015) to track user requests. By taking each user input as new evidence, the task of a belief tracker is to maintain a multinomial distribution p over values $v \in V_s$ for each informable slot[2] s, and a binary distribution for each requestable slot[2]. These probability distributions $\mathbf{p}_t^s$ are called belief states of the system. The belief states $\mathbf{p}_t^s$, together with the intent vector $\mathbf{z}_t$, can be viewed as the system's comprehension of the user requests up to turn t.

Database Operator Based on the belief states $\mathbf{p}_t^s$, a DB query is formed by taking the union of the maximum values of each informable slot. A vector $\mathbf{x}_t$ representing different degrees of matching in the DB (no match, 1 match, ... or more than 5 matches) is produced by counting the number of matched entities and expressing it as a 6-bin 1-hot encoding. If $\mathbf{x}_t$ is not zero, an associated entity pointer is maintained which identifies one of the matching DB entities selected at random. The entity pointer is updated if the current entity no longer matches the search criteria; otherwise it stays the same.

Policy Network Based on the vectors $\mathbf{z}_t$, $\mathbf{p}_t^s$, and $\mathbf{x}_t$ from the above three modules, the policy network combines them into a single action vector $\mathbf{m}_t$ by a three-way matrix transformation,

$$\mathbf{m}_t = \tanh(\mathbf{W}_{zm}\mathbf{z}_t + \mathbf{W}_{xm}\mathbf{x}_t + \textstyle\sum_{s\in\mathbb{G}} \mathbf{W}_{pm}^s \mathbf{p}_t^s) \quad (1)$$

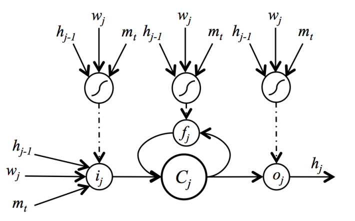
where matrices $\mathbf{W}_{zm}$, $\mathbf{W}_{pm}^s$, and $\mathbf{W}_{xm}$ are parameters and $\mathbb{G}$ is the domain ontology.

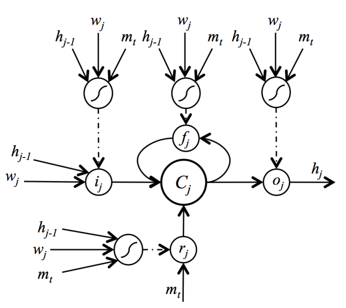
3.2 Decoder Module

Conditioned on the system action vector $\mathbf{m}_t$ provided by the encoder module, the decoder module uses a conditional LSTM LM to generate the required system output token by token in skeletal form[1]. The final system response can then be formed

[1]Delexicalisation: slots and values are replaced by generic tokens (e.g. keywords like *Chinese food* are replaced by *[v.food] [s.food]* to allow weight sharing.

[2]Informable slots are slots that users can use to constrain the search, such as food type or price range; Requestable slots are slots that users can ask a value for, such as phone number. This information is specified in the domain ontology.


(a) Language model type LSTM


(b) Memory type LSTM

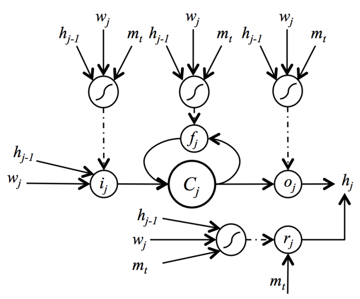

(c) Hybrid type LSTM

Figure 1: Three different conditional generation architectures.

by substituting the actual values of the database entries into the skeletal sentence structure.

3.2.1 Conditional Generation Network

In this paper we study and analyse three different variants of LSTM-based conditional generation architectures:

Language Model Type The most straightforward way to condition the LSTM network on additional source information is to concatenate the conditioning vector $\mathbf{m}_t$ together with the input word embedding $\mathbf{w}_j$ and previous hidden layer $\mathbf{h}_{j-1}$,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \hat{\mathbf{c}}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \tanh \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{m}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j)$$

where index j is the generation step, n is the hidden layer size, $\mathbf{i}_j, \mathbf{f}_j, \mathbf{o}_j \in [0,1]^n$ are input, forget, and output gates respectively, $\hat{\mathbf{c}}_j$ and $\mathbf{c}_j$ are proposed cell value and true cell value at step j, and $\mathbf{W}_{4n,3n}$ are the model parameters. The model is shown in Figure 1a. Since it does not differ significantly from the original LSTM, we call it the *language model type* (lm) conditional generation network.

Memory Type The *memory type* (mem) conditional generation network was introduced by Wen et al. (2015b), shown in Figure 1b, in which the conditioning vector $\mathbf{m}_t$ is governed by a standalone reading gate $\mathbf{r}_j$. This reading gate decides how much information should be read from the conditioning vector and directly writes it into the memory cell $\mathbf{c}_j$,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \mathbf{r}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{m}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\hat{\mathbf{c}}_j = \tanh\left(\mathbf{W}_c(\mathbf{w}_j \oplus \mathbf{h}_{j-1})\right)$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j + \mathbf{r}_j \odot \mathbf{m}_t$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j)$$

where $\mathbf{W}_c$ is another weight matrix to learn. The idea behind this is that the model isolates the conditioning vector from the LM so that the model has more flexibility to learn to trade off between the two.

Hybrid Type Continuing with the same idea as the *memory type* network, a complete separation of conditioning vector and LM (except for the gate controlling the signals) is provided by the *hybrid type* network shown in Figure 1c,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \mathbf{r}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{m}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\hat{\mathbf{c}}_j = \tanh\left(\mathbf{W}_c(\mathbf{w}_j \oplus \mathbf{h}_{j-1})\right)$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j) + \mathbf{r}_j \odot \mathbf{m}_t$$

This model was motivated by the fact that long-term dependency is not needed for the conditioning vector because we apply this information at every step j anyway. The decoupling of the conditioning vector and the LM is attractive because it leads to better interpretability of the results and provides the potential to learn a better conditioning vector and LM.

3.2.2 Attention and Belief Representation

Attention An attention-based mechanism provides an effective approach for aggregating multiple information sources for prediction tasks. Like Wen et al.

(2016a), we explore the use of an attention mechanism to combine the tracker belief states in which the policy network in Equation 1 is modified as

$$\mathbf{m}_t^j = \tanh(\mathbf{W}_{zm}\mathbf{z}_t + \mathbf{W}_{xm}\mathbf{x}_t + \textstyle\sum_{s\in\mathbb{G}} \alpha_s^j \mathbf{W}_{pm}^s \mathbf{p}_t^s)$$

where the attention weights α_s^j are calculated by,

$$\alpha_s^j = \mathrm{softmax}\big(\mathbf{r}^\intercal \tanh\big(\mathbf{W}_r \cdot (\mathbf{v}_t \oplus \mathbf{p}_t^s \oplus \mathbf{w}_j^t \oplus \mathbf{h}_{j-1}^t)\big)\big)$$

where $\mathbf{v}_t = \mathbf{z}_t + \mathbf{x}_t$ and matrix $\mathbf{W}_r$ and vector $\mathbf{r}$ are parameters to learn.

Belief Representation The effect of different belief state representations on the end performance are also studied. For user informable slots, the *full* belief state $\mathbf{p}_t^s$ is the original state containing all categorical values; the *summary* belief state contains only three components: the summed value of all categorical probabilities, the probability that the user said they "don't care" about this slot and the probability that the slot has not been mentioned. For user requestable slots, on the other hand, the full belief state is the same as the summary belief state because the slot values are binary rather than categorical.

3.3 Snapshot Learning

Learning conditional generation models from sequential supervision signals can be difficult, because it requires the model to learn both long-term word dependencies and potentially distant source encoding functions. To mitigate this difficulty, we introduce a novel method called *snapshot learning* to create a vector of binary labels $\boldsymbol{\Upsilon}_t^j \in [0,1]^d$, $d < dim(\mathbf{m}_t^j)$ as the *snapshot* of the remaining part of the output sentence $T_{t,j:|T_t|}$ from generation step j. Each element of the snapshot vector is an indicator function of a certain event that will happen in the future, which can be obtained either from the system response or dialogue context at training time. A *companion* cross entropy error is then computed to force a subset of the conditioning vector $\hat{\mathbf{m}}_t^j \subset \mathbf{m}_t^j$ to be close to the snapshot vector,

$$L_{ss}(\cdot) = -\textstyle\sum_t \sum_j \mathbb{E}[H(\Upsilon_t^j, \hat{m}_t^j)] \qquad (2)$$

where $H(\cdot)$ is the cross entropy function, Υ_t^j and $\hat{m}_t^j$ are elements of vectors $\boldsymbol{\Upsilon}_t^j$ and $\hat{\mathbf{m}}_t^j$, respectively. In order to make the $\tanh$ activations of $\hat{\mathbf{m}}_t^j$ compatible with the 0-1 snapshot labels, we squeeze each

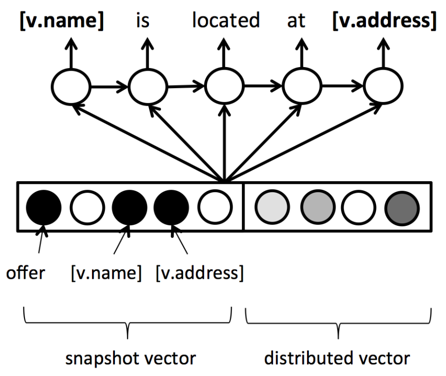


Figure 2: The idea of snapshot learning. The snapshot vector was trained with additional supervisions on a set of indicator functions heuristically labelled using the system response.

value of $\hat{\mathbf{m}}_t^j$ by adding 1 and dividing by 2 before computing the cost.

The indicator functions we use in this work have two forms: (1) whether a particular slot value (e.g., *[v.food]*[1]) is going to occur, and (2) whether the system has offered a venue[3], as shown in Figure 2. The *offer* label in the snapshot is produced by checking the delexicalised name token (*[v.name]*) in the entire dialogue. If it has occurred, every label in subsequent turns is labelled with 1. Otherwise it is labelled with 0. To create snapshot targets for a particular slot value, the output sentence is matched with the corresponding delexicalised token turn by turn, per generation step. At each generation step, the target is labelled with 0 if that delexicalised token has been generated; otherwise it is set to 1. However, for the models without attention, the targets per turn are set to the same because the condition vector will not be able to learn the dynamically changing behaviour without attention.

4 Experiments

Dataset The dataset used in this work was collected in the Wizard-of-Oz online data collection described by Wen et al. (2016a), in which the task of the system is to assist users to find a restaurant in Cambridge, UK area. There are three informable slots (*food*, *pricerange*, *area*) that users can use to constrain the search and six requestable slots (*address*, *phone*, *postcode* plus the three informable

[3]Details of the specific application used in this study are given in Section 4 below.

Architecture	Belief	Success(%)	SlotMatch(%)	T5-BLEU	T1-BLEU
Belief state representation					
lm	full	72.6 / 74.5	52.1 / 60.3*	0.207 / 0.229*	0.216 / 0.238*
lm	summary	74.5 / 76.5	57.4 / 61.2*	0.221 / 0.231*	0.227 / 0.240*
Conditional architecture					
lm	summary	74.5 / 76.5	57.4 / 61.2*	0.221 / 0.231*	0.227 / 0.240*
mem	summary	75.5 / 77.5	59.2 / **61.3***	0.222 / **0.232***	0.231 / **0.243***
hybrid	summary	76.1 / 79.2	52.4 / 60.6*	0.202 / 0.228*	0.212 / 0.237*
Attention-based model					
lm	summary	79.4 / 78.2	60.6 / 60.2	0.228 / 0.231	0.239 / 0.241
mem	summary	76.5 / 80.2*	57.4 / 61.0*	0.220 / 0.229	0.228 / 0.239
hybrid	summary	79.0 / **81.8***	56.2 / 60.5*	0.214 / 0.227*	0.224 / 0.240*

Table 1: Performance comparison of different model architectures, belief state representations, and snapshot learning. The numbers to the left and right of the / sign are learning without and with snapshot, respectively. The model with the best performance on a particular metric (column) is shown in bold face. The *lm* models in *Conditional architecture* and *Attention-based model* are the same models as in Wen et al. (2016a). Statistical significance was computed using two-tailed Wilcoxon Signed-Rank Test (* p <0.05) to compare models w/ and w/o snapshot learning.

slots) that the user can ask a value for once a restaurant has been offered. There are 676 dialogues in the dataset (including both finished and unfinished dialogues) and approximately 2750 turns in total. The database contains 99 unique restaurants.

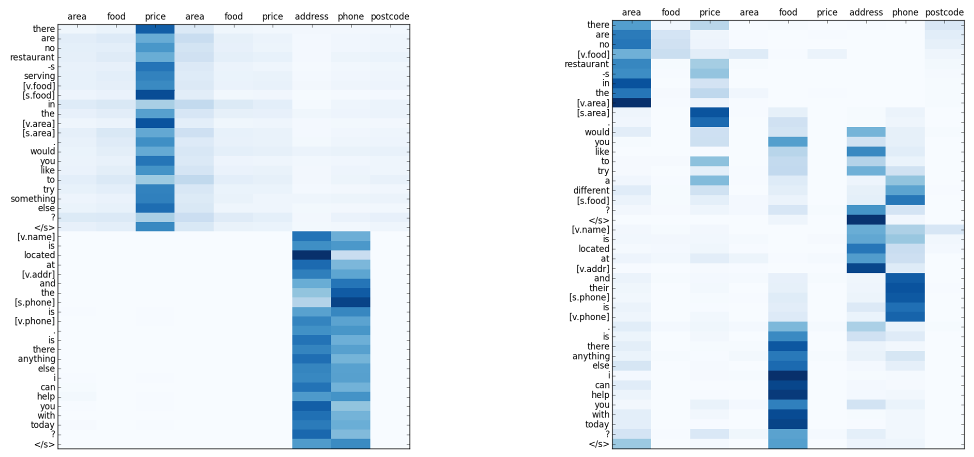
Training The training procedure was divided into two stages. Firstly, the belief tracker parameters θ_b were pre-trained using cross entropy errors between tracker labels and predictions. Having fixed the tracker parameters, the remaining parts of the model $\theta_{\backslash b}$ are trained using the cross entropy errors from the generation network LM,

$$L(\theta_{\backslash b}) = -\sum_t \sum_j H(\mathbf{y}_j^t, \mathbf{p}_j^t) + \lambda L_{ss}(\cdot) \tag{3}$$

where $\mathbf{y}_j^t$ and $\mathbf{p}_j^t$ are output token targets and predictions respectively, at turn t of output step j, $L_{ss}(\cdot)$ is the snapshot cost from Equation 2, and λ is the tradeoff parameter in which we set to 1 for all models trained with snapshot learning. We treated each dialogue as a batch and used stochastic gradient descent with a small $l2$ regularisation term to train the model. The collected corpus was partitioned into a training, validation, and testing sets in the ratio 3:1:1. Early stopping was implemented based on the validation set considering only LM log-likelihoods. Gradient clipping was set to 1. The hidden layer sizes were set to 50, and the weights were randomly initialised between -0.3 and 0.3 including word embeddings. The vocabulary size is around 500 for both input and output, in which rare words and words that can be delexicalised have been removed.

Decoding In order to compare models trained with different recipes rather than decoding strategies, we decode all the trained models with the average log probability of tokens in the sentence. We applied beam search with a beamwidth equal to 10, the search stops when an end-of-sentence token is generated. In order to consider language variability, we ran decoding until 5 candidates were obtained and performed evaluation on them.

Metrics We compared models trained with different recipes by performing a corpus-based evaluation in which the model is used to predict each system response in the held-out test set. Three evaluation metrics were used: BLEU score (on top-1 and top-5 candidates) (Papineni et al., 2002), slot matching rate and objective task success rate (Su et al., 2015). The dialogue is marked as successful if both: (1) the offered entity matches the task that was specified to the user, and (2) the system answered all the associated information requests (e.g. *what is the address?*) from the user. The slot matching rate is the percentage of delexicalised tokens (e.g. *[s.food]* and *[v.area]*[1]) appear in the candidate also appear in the



(a) *Hybrid* LSTM w/o snapshot learning (b) *Hybrid* LSTM w/ snapshot learning

Figure 3: Learned attention heat maps over trackers. The first three columns in each figure are informable slot trackers and the rest are requestable slot trackers. The generation model is the *hybrid* type LSTM.

reference. We computed the BLEU scores on the skeletal sentence forms before substituting with the actual entity values. All the results were averaged over 10 random initialised networks.

Results Table 1 shows the evaluation results. The numbers to the left and right of each table cell are the same model trained w/o and w/ snapshot learning. The first observation is that snapshot learning consistently improves on most metrics regardless of the model architecture. This is especially true for BLEU scores. We think this may be attributed to the more discriminative conditioning vector learned through the snapshot method, which makes the learning of the conditional LM easier.

In the first block *belief state representation*, we compare the effect of two different belief representations. As can be seen, using a succinct representation is better (*summary*>*full*) because the identity of each categorical value in the belief state does not help when the generation decisions are done in skeletal form. In fact, the full belief state representation may encourage the model to learn incorrect coadaptation among features when the data is scarce.

In the *conditional architecture* block, we compare the three different conditional generation architectures as described in section 3.2.1. This result shows that the language model type (*lm*) and memory type (*mem*) networks perform better in terms of BLEU score and slot matching rate, while the hybrid type (*hybrid*) networks achieve higher task success. This is probably due to the degree of separation between the LM and conditioning vector: a coupling approach (*lm, mem*) sacrifices the conditioning vector but learns a better LM and higher BLEU; while a complete separation (*hybrid*) learns a better conditioning vector and offers a higher task success.

Model	$\mathbf{i}_j$	$\mathbf{f}_j$	$\mathbf{r}_j/\mathbf{o}_j$
hybrid, full	0.567	0.502	0.405
hybrid, summary	0.539	0.540	0.428
+ att.	0.540	0.559	0.459

Table 2: Average activation of gates on test set.

Lastly, in the *attention-based model* block we train the three architectures with the attention mechanism and compare them again. Firstly, the characteristics of the three models we observed above also hold for attention-based models. Secondly, we found that the attention mechanism improves all the three architectures on task success rate but not BLEU scores. This is probably due to the limitations of using n-gram based metrics like BLEU to evaluate the generation quality (Stent et al., 2005).

5 Model Analysis

Gate Activations We first studied the average activation of each individual gate in the models by averaging them when running generation on the test set. We analysed the *hybrid* models because their reading gate to output gate activation ratio ($\mathbf{r}_j/\mathbf{o}_j$) shows clear tradeoff between the LM and the conditioning vector components. As can be seen in Ta-

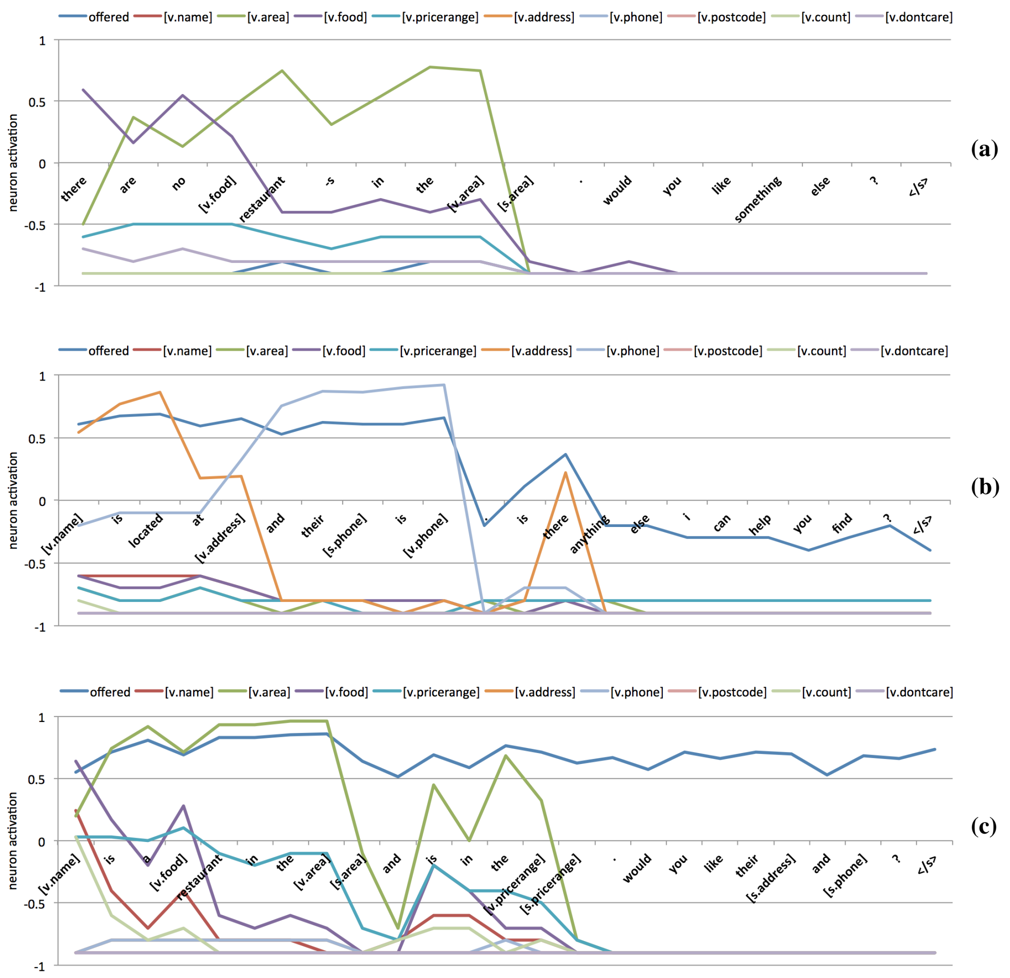


Figure 4: Three example responses generated from the hybrid model trained with snapshot and attention. Each line represents a neuron that detects a particular snapshot event.

ble 2, we found that the average forget gate activations ($\mathbf{f}_j$) and the ratio of the reading gate to the output gate activation ($\mathbf{r}_j/\mathbf{o}_j$) have strong correlations to performance: a better performance (*row 3>row 2>row 1*) seems to come from models that can learn a longer word dependency (higher forget gate $\mathbf{f}_t$ activations) and a better conditioning vector (therefore higher reading to output gate ratio $\mathbf{r}_j/\mathbf{o}_j$).

Learned Attention We have visualised the learned attention heat map of models trained with and without snapshot learning in Figure 3. The attention is on both the informable slot trackers (first three columns) and the requestable slot trackers (the other columns). We found that the model trained with snapshot learning (Figure 3b) seems to produce a more accurate and discriminative attention heat map comparing to the one trained without it (Figure 3a). This may contribute to the better performance achieved by the snapshot learning approach.

Snapshot Neurons As mentioned earlier, snapshot learning forces a subspace of the conditioning vector $\hat{\mathbf{m}}_t^j$ to become discriminative and interpretable. Three example generated sentences together with the snapshot neuron activations are shown in Figure 4. As can be seen, when generating words one by one, the neuron activations were changing to detect different events they were assigned by the snapshot training signals: e.g. in Figure 4b the *light blue* and *orange* neurons switched their domination role when the token *[v.address]* was generated; the *offered* neuron is in a high activation state in Figure 4b because the system was offering a venue, while in Figure 4a it is not activated because the system was still helping the user to find a venue.

6 Conclusion and Future Work

This paper has investigated different conditional generation architectures and a novel method called snapshot learning to improve response generation in a neural dialogue system framework. The results showed three major findings. Firstly, although the *hybrid type* model did not rank highest on all metrics, it is nevertheless preferred because it achieved the highest task success and also it provided more interpretable results. Secondly, snapshot learning provided gains on virtually all metrics regardless of the architecture used. The analysis suggested that the benefit of snapshot learning mainly comes from the more discriminative and robust subspace representation learned from the heuristically labelled companion signals, which in turn facilitates optimisation of the final target objective. Lastly, the results suggested that by making a complex system more interpretable at different levels not only helps our understanding but also leads to the highest success rates.

However, there is still much work left to do. This work focused on conditional generation architectures and snapshot learning in the scenario of generating dialogue responses. It would be very helpful if the same comparison could be conducted in other application domains such as machine translation or image caption generation so that a wider view of the effectiveness of these approaches can be assessed. Furthermore, removing slot-value delexicalisation and learning confirmation behaviour in noisy speech conditions are also main research problems from the system development prospective.

Acknowledgments

Tsung-Hsien Wen and David Vandyke are supported by Toshiba Research Europe Ltd, Cambridge Research Laboratory.

References

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *ICLR*.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *ICASSP*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint:1308.0850*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *SIGdial*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

John F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transaction on Information Systems*.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *AISTATS*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. *arXiv perprint:1603.06155*.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and

Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint:1603.06744*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGdial*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL*.

Tomáš Mikolov, Martin Karafit, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *InterSpeech*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan H. Černocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015a. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint:1512.05742*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015b. Hierarchical neural network generative models for movie dialogues. In *AAAI*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *CICLing 2005*.

Pei-Hao Su, David Vandyke, Milica Gasic, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve J. Young. 2015. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Interspeech*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

David R. Traum, 1999. *Foundations of Rational Agency*, chapter Speech Acts for Dialogue Agents. Springer.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGdial*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016a. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint:1604.04562*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2016b. Multi-domain neural network language generation for spoken dialogue systems. In *NAACL-HLT*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables. In *IJCAI*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer, Speech and Language*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of IEEE*.