Master of Science in Industrial and Applied Mathematics (MSIAM)
Specialization: Data Science (2016-17)

Character-based Recurrent Neural Networks for Natural Language Generation from Dialogue Acts

Shubham Agarwal

19 June, 2017

Research project performed at:
XRCE (Xerox Research Center Europe) Grenoble, France

Under the supervision of:
Dr. Marc DYMETMAN, Principal Scientist, XRCE
Prof. Eric GAUSSIER, Director, LIG, Université Grenoble Alpes

Confidential

Abstract

Natural Language Generation in Dialogue Acts involves generating human understandable utterances from slot-value pairs in Meaning Representation (MR). This forms an integrated component in Spoken Dialogue Systems, where the recent advances in Deep Learning are shaping the research towards using end to end models. We trained a char2char model on the recently released *E2E NLG Challenge* data, which compared to existing datasets has an open vocabulary, complex syntactic structures and diverse discourse phenomena.

With minimal effort, and in particular without delexicalization, tokenization and even lowercasing, the obtained raw predictions, according to a small scale human evaluation, are excellent on the linguistic side and quite reasonable on the adequacy side, the primary downside being the possible omissions of semantic material. However, in a significant number of cases (more than 70%), a perfect solution can be found in the top-20 predictions, indicating promising directions for solving the remaining issues.

Similar to the re-ranking approaches in Machine Translation, we explored two different approaches to score candidates in the top k predictions which directly impacts our future submission to this challenge. During the course of our thesis, we also submitted a paper to SIGdial 2017 which recently got accepted.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Dr. Marc DYMETMAN for allowing this wonderful opportunity to work in the field of Natural Language Processing. His invaluable assistance and comments helped me understand more about the problem statement, the accompanying challenges and techniques to tackle with Natural Language Generation. His efforts made possible the accepted submission to SIGdial 2017 conference. Prof. Eric GAUSSIER provided valuable insights and recommendations from time to time. Their joint supervision guided me throughout the course of my thesis and would undoubtably influence our intended submission to the E2E NLG Challenge.

Contents

— 1 —
Introduction

Recently, Spoken Dialogue Systems (SDS) have gained immense attraction in the research community, wherein a user interacts with machines for a particular task. Natural Language Generation (NLG) is an important component of these Spoken Dialogue Systems (SDS) critically impacting user experience and thus affecting user retention. This module takes as input a *Meaning Representation* (also commonly referred as 'Dialogue Act' in literature) and tries to generate human understandable utterances relevant to the task. More information about the Spoken Dialogue Systems and Natural Language Generation aspect is covered in Section 3.

Very recently, researchers (Novikova et al., 2017) at Heriot-Watt University proposed the E2E NLG Challenge[1] and released a dataset consisting of 50K (MR, RF) pairs, MR being a slot-value Meaning Representation of a restaurant, RF (human ReFerence) being a natural language utterance rendering of that representation. The utterances were crowd-sourced based on pictorial representations of the MRs, with the intention of producing more natural and diverse utterances compared to the ones directly based on the original MRs (Novikova et al., 2016). Compared to the earlier available small datasets, this challenge introduces additional complexity of open vocabulary, complex syntactic structures and diverse discourse phenomena for language generation. The slots (a.k.a 'attributes' in dialogue literature) are canonically ordered in this dataset, having a systematic structure with a total of 10 slot types. We provide a more detailed description of the dataset in Section 5. Our work during the course of this thesis is motivated by a planned submission to this challenge.

We used Character based Sequence to Sequence (commonly termed as Encoder Decoder Recurrent Neural Networks (RNN)) models (Sutskever et al., 2014; Cho et al., 2014) with attention mechanism (Bahdanau et al., 2014), whereby we feed character embeddings of the source Meaning Representation (MR) to the Encoder RNN and try to predict the character sequences of the target RF (crowd-sourced utterances) in the generation stage by the Decoder RNN. This kind of model is standardly trained using 'Teacher Forcing' algorithm where actual target values are used as input at a future time step to Decoder RNN, which can be different from the output at previous time steps. During decoding we compare greedy search with sampling as well as beam search based inference. Without any pre- or post-processing (not even tokenization or lowercasing), we obtain excellent predictions on which we conducted a small-scale human evaluation on one hundred MRs, involving two evaluators. This evaluation, on one

[1]`http://www.macs.hw.ac.uk/InteractionLab/E2E/`

hand, concentrated on the linguistic quality, and on the other, semantic adequacy of the produced utterances. On the linguistic side, the vast majority of the predictions were surprisingly grammatically perfect, while still being rather diverse and natural. In particular, our char-based model never produced non-words. On the adequacy side, we found that the only serious problem was the tendency (in about half of the evaluated cases) of the model to omit to render one (rarely two) slot(s); on the other end, it never hallucinated, and very rarely duplicated, material.

There is a current focus in Machine Translation towards re-ranking the top-k predictions (*n-best list*) produced using beam search for each input sequence. To try and assess the potential value of a simple re-ranking technique for our use case in dialogue, we generated, using beam-search, 20-best utterances for each MR. The evaluators also scanned towards finding an *oracle*, i.e. a generated utterance considered as perfect not only from the grammatical but also from the adequacy viewpoint. An oracle was found in the first position in around 50% of the case, otherwise among the 20 positions in around 20% of the cases, and not at all inside this list in the remaining 30% cases. On the basis of these experiments and evaluations we believe that there is only a modest gap towards a very reasonable NLG seq2seq based model for the challenge.

As a first approach in this direction, we experimented with a procedure similar to 'inverted generation' technique by (Chisholm et al., 2017). We generated top-k list of predictions (using Beam Search) for each MR in what we call the *forward* phase of the model. In parallel, we trained a *reverse* model which tries to reconstruct the MR given the target RF. This is guided by an intuition that if our prediction omits some information, the reverse reconstruction of MR would also tend to omit slot-value pairs for the omitted slot values in the prediction. We might then score and re-rank the top-k predictions based on a distance metric such as Edit Distance between the MR and the generated representations in reverse direction of the predictions (produced in the forward direction). For a superficial evaluation using the automatic metric of BLEU, we compared the re-ranked predictions with the best predictions (most probable prediction) produced using beam search, but the metric deteriorated marginally. We will discuss more about this architecture in Section 4.

To treat omission (generalized to any kind of *semantic adequacy* mis-representation such as repetition or addition of content) in the predictions as a classification task, we developed a dataset using a protocol defined in Section 4. We propose Siamese-kind architecture (Bromley et al., 1994; Mueller and Thyagarajan, 2016) as an alternate approach to detect omission (semantic adequacy in general) and hence re-rank the top-k predictions for each MR. At the time of writing of thesis, we are at the initial stage of developing this novel architecture, which we hope can further improve our model predictions.

The rest of our work is organized as follows. We summarize related work in Section 2 and begin by introducing the theoretical background in Section 3. In Section 4, we elaborate more about our model, providing relevant statistics as well as the implementation procedure in Section 5. Section 6 provides a general overview of our experiments and finally Section 7 will give an outline of future work. We also attach a blind copy of our submission to SIGdial at the end.

— 2 —
Related Work

Traditionally, the Natural Language Generation (NLG) component in Spoken Dialogue Systems have been rule-based systems involving a two stage pipeline: ‘sentence planning’ or ‘content selection’ (deciding the overall structure of sentence) and ‘surface realization’ which renders actual utterances using this structure. The resulting utterances using these rule based systems tends to be rigid, repetitive and limited in scope. NLG is a complex structured prediction task with the set of possible output predictions forming a high dimensional space. Recent approaches in dialogue generation tend to directly learn the predictions from un-aligned data (Mei et al., 2015; Lampouras and Vlachos, 2016; Dušek and Jurčíček, 2016; Wen et al., 2015).

Recurrent Neural Networks and their variants such as LSTMs and GRUs (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) are now extensively used in Natural Language Processing, capitalizing on their ability to model sequential data where we treat the input text as a sequence of words (sometimes as characters). This class of neural networks when integrated in a Sequence to Sequence (Cho et al., 2014; Sutskever et al., 2014) framework, have produced state-of-art results in Machine Translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014), Text (abstract) Summarization (Rush et al., 2015), Conversational Modeling (Vinyals and Le, 2015) and Semantic Parsing (Xiao et al., 2016). With some variant in the encoder (sometimes using CNN) these kind of models have also been utilized for Image Captioning (Vinyals et al., 2015; Karpathy and Fei-Fei, 2015) and Video Captioning (Venugopalan et al., 2015). While these models were initially developed to be used at word level in NLP related tasks, there has been a recent interest to use character level sequences as in Machine Translation (Chung et al., 2016; Zhao and Zhang, 2016; Ling et al., 2015b).

Most of the RNN-based approaches to Natural Language Generation (NLG) that we are aware of, starting with (Wen et al., 2015), generate the output word-by-word, and resort to special delexicalization (process in which they replace named entities (slot values) with special ‘placeholders’) or copy mechanisms (Gu et al., 2016) to handle rare or unknown words, for instance restaurant names or telephone numbers. One exception is (Goyal et al., 2016), who employs a char-based seq2seq model where the input MR is simply represented as a character sequence, and the output is also generated char-by-char; this approach avoids the rare word problem, as the character vocabulary is very small.

Recently, a team at Google Brain open sourced 'tf-seq2seq' [1], a general encoder-decoder framework based on Tensorflow (Abadi et al., 2016), provided along with (Britz et al., 2017), with some standard configuration options that will be detailed in Section 5. While in their large-scale NMT experiments (Britz et al., 2017) use word-based sequences, in our case we use character-based ones. Our initial experiments using this framework were guided by the recommended configurations in Table 7 of (Britz et al., 2017). Even though performed on a completely different task for Machine Translation using word based models, we found a large correspondence with these configurations for our use case of NLG using Character based models which we describe more in Section 6.

Inspired by (Goyal et al., 2016) and using a similar approach, we experimented with different numbers of layers in the encoder and decoder as well as different beam widths, while using the bi-directional encoder along with the 'additive' attention mechanism. While (Goyal et al., 2016) used an additional finite-state mechanism to guide the production of well-formed (and input-motivated) character sequences, our char-based model never produced non-words, contrary to their findings. This can probably be attributed to the nature of our comparatively larger E2E NLG dataset which in contrast to the dataset used by (Goyal et al., 2016), only contain a small number of different entity names (restaurant names, location) and no addresses or telephone numbers. Because of the diversity of these entities in their dataset, it is difficult for their model to generate them correctly without the guidance of the finite state mechanisms. Compared to them, our model is *deeper* with more depth in both encoder and decoder. We used character embeddings instead of one-hot encodings and non-null 'length-penalty' (alias length normalization (Wu et al., 2016)) when using beam search for inference. As also observed by (Britz et al., 2017), using a non-zero length-penalty, significantly improved the decoding results, giving a minimum uplift of 2 in the BLEU score for the same beam width (see Table 6.1). We also implemented temperature-sampling based methods for decoding over this framework. Though producing more diverse utterances and better BLEU scores compared to greedy search, they were constantly outperformed by beam search based inference when we use a length penalty of value 1.

For re-ranking, as described in Section 1, we use an approach similar to 'inverted generation' technique of (Chisholm et al., 2017). We train our model in reverse direction to generate reconstruction of MRs from the target RFs. The dissimilarity between the original and reconstructed MR for the output predictions are used to provide scores for re-ranking. The classifier used to detect omission and addition (our alternate approach defined in Section 1) is based upon the siamese architectures described in (Mueller and Thyagarajan, 2016; Neculoiu et al., 2016).

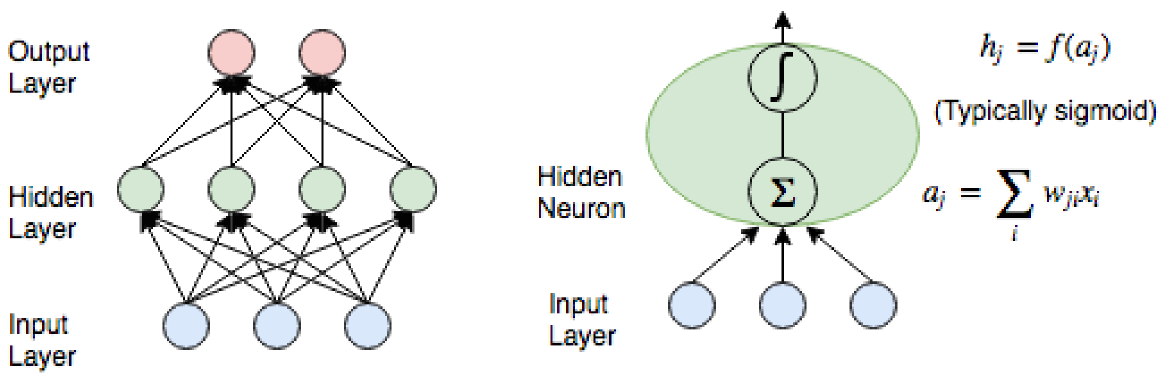
[1] `https://github.com/google/seq2seq`.

— 3 —
Theoretical Background

This chapter covers the mathematical background of our approach. From a small recap of feedforward neural networks, we move over to Recurrent Neural Networks and their more generally used variants such as LSTMs and GRUs. We will also introduce the sequence to sequence networks and their training procedure. Lastly, we discuss Spoken Dialogue Systems and the Natural Language Generation approaches towards these systems.

3.1 Feedforward Neural Networks

Feedforward neural networks are the basic building blocks in the field of Deep Learning. Typically, a neural network has an input layer and one or more hidden layers which try to model the predictions at the output layer as depicted in Fig. 3.1.



(a) (b)

Figure 3.1: (a) Feedforward neural network. (b) activation at a (hidden) neuron. Non-linearity (here we depicted sigmoid) can also vary across layers. These networks are trained by the popular Backpropagation algorithm.

Given an input $x \in \mathbb{R}^n$ and the corresponding output $y \in \mathbb{R}^m$, a neural network parametrized by ($W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$) generates the prediction using Equation 3.1 where f is a point-wise nonlinearity (typically sigmoid or tanh).

$$y_{pred} = f(Wx + b) \tag{3.1}$$

We also calculate the loss between the target output y and the predicted outputs y_{pred}, often cross entropy in classification related tasks. Based on this loss, the parameters of this model are trained by a stochastic gradient descent algorithm, typically Back-propagation (Hecht-Nielsen, 1989).

A simple Feedforward Neural Network considers the input data to be of fixed size and independent across each feature domain. In practice, however, input features have temporal dependencies exhibiting sequential nature. Recurrent Neural Networks described in the next section are better suited to model these sequential data inputs.

3.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (Elman, 1990), to some extent, can be considered as the *de facto* architectures while dealing with sequential data. They also find great applications in NLP related tasks for language modelling as well as classification.

3.2.1 Vanilla RNNs

Typically, a vanilla RNN takes an input sequence $x = (x_1, x_2, \ldots, x_t, \ldots, x_n)$ and models it using hidden vectors with shared weight parameters (see Equation 3.2 and 3.3, here we use tanh as the non-linearity over the weighted combination to calculate the hidden state vector). The hidden state of the RNN acts as a *memory*, capturing this temporal information from the input sequences.

$$\begin{aligned} h_t &= f_W(h_{t-1}, x_t) \\ &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_{(h)}) \end{aligned} \tag{3.2}$$

$$\begin{aligned} y_t &= g_W(h_t) \\ &= \text{softmax}(W_{hy}h_t + b_{(y)}) \end{aligned} \tag{3.3}$$

Here we have $x_t \in R^n, h_t \in R^m, W_{hh} \in R^{m\times m}, W_{xh} \in R^{m\times n}$ and f_W as the point wise non-linearity (showed as tanh). $b_{(y)}$ and $b_{(h)}$ are bias terms. We transform this hidden vector in the output space by calculating softmax probabilities (which performs normalized exponential over the activation scores, transforming them into a probability distribution), given by:

$$\text{softmax}(z_j) = \frac{exp(z_j)}{\sum_{k=1}^{K} exp(z_k)} \tag{3.4}$$

The total number of parameters for this kind of model is thus $2nm + m^2$. Typically, the input sequence to a RNN is encoded as 'one-hot representation', but we can also have the input represented by an embedding layer. Compared to traditionally used Hidden Markov Models (HMMs) for sequential tasks, RNNs can be used to model sequential dependency over a larger number of time steps (though with refined structure variants defined in Section 3.2.3).

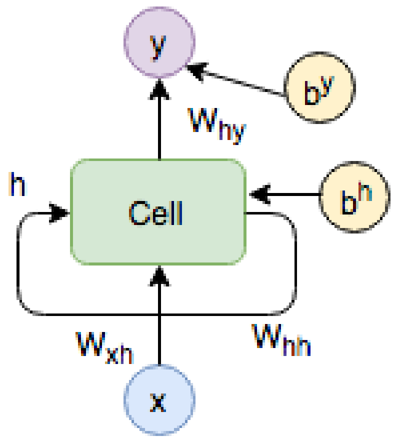


Figure 3.2: A simple RNN unit

In contrast to traditional Deep (Feed-forward) Neural Networks, all the weight parameters (W_{hh}, W_{xh}, W_{hy}) and bias terms $(b_{(h)}, b_{(y)})$ are shared at each time step. Due to these cyclic dependencies, RNNs are generally considered as generalization of Feed forward networks when 'unfolded' in time domain (Fig. 3.3) To take into account this additional time dimension, one uses an updated version of Backpropagation algorithm to train these networks, called Back Propagation Through Time (BPTT) (Werbos, 1990).

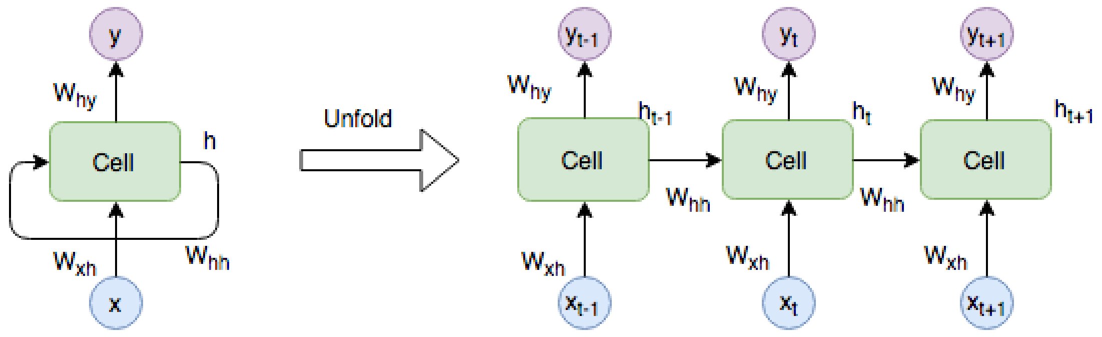


Figure 3.3: Unfolded RNN trained using BPTT. (Here we do not show the bias term for simplicity)

Theoretically, RNNs can exploit information in arbitrary long sequences. But they suffer from the issue of vanishing gradients (Bengio et al., 1994) as discussed more in Section 3.2.3.

3.2.2 Bi-directional RNNs

Bi-directional RNNs (Schuster and Paliwal, 1997) are a variant of Recurrent Neural Networks, where we have two RNNs, forward and backward running in parallel for the same sequential input. This is inspired by the idea that the information about the output at each time step is dependent upon not only the previous input but also upon the future sequence. They have shown promising results in Named Entity Recognition (Huang et al., 2015; Ling et al., 2015a) and standardly employed as Encoder in Sequence to Sequence learning with attention models (Bahdanau et al., 2014) defined in Section 3.3. Gated RNNs defined in Section 3.2.3, can also be used in this kind of framework, instead of vanilla RNNs. Given an input sequence $x = (x_1, x_2, \ldots, x_t, \ldots, x_n)$ and the hidden states for forward and backward represented by h_t and z_t, the output of cell at each time step is given by

$$h_t = f_w(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{3.5}$$

$$z_t = f_z(W_{zx}z_t + W_{zz}z_{t-1} + b_z) \tag{3.6}$$

$$y_t = \text{softmax}(W_{yh}h_t + W_{yz}z_t + b_y) \tag{3.7}$$

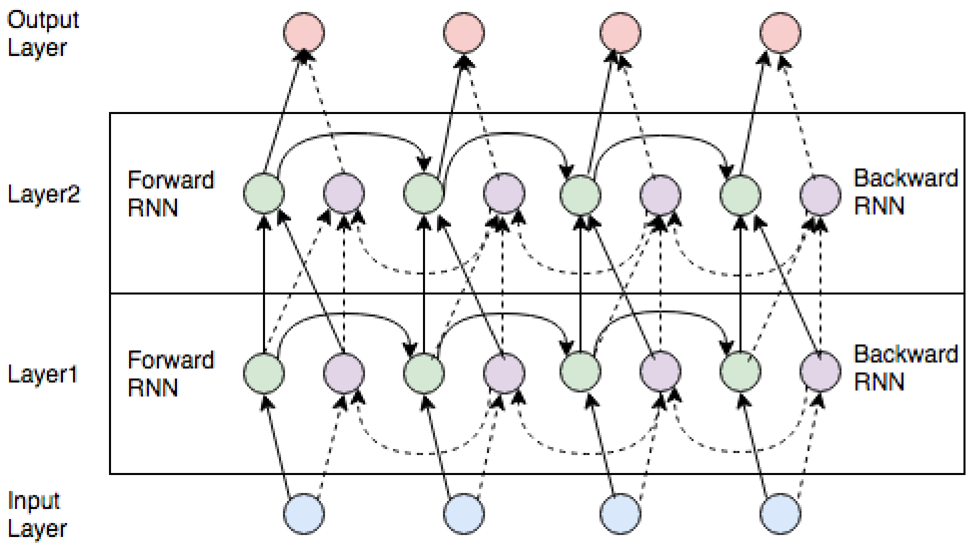


Figure 3.4: Deep Bi-directional RNN. Here we show the architecture with two layers and four input time steps for simplicity. (Forward RNN is marked in green and Backward RNN in purple. Input to backward RNN are marked by dotted lines while forward RNN inputs are depicted by solid lines)

3.2.3 Gated RNNs: LSTMs and GRUs

As described in Section 3.2, vanilla RNNs are not able to capture long term dependencies because of the issue of vanishing gradients, whereby the gradient value decreases exponentially while reaching layers farther in time. (Hochreiter and Schmidhuber, 1997) introduced LSTMs specifically to address this.

The core idea behind LSTMs is the gating mechanism which gives the ability to add or remove information to the cell state, optionally letting information through. The usual cell of Vanilla RNN is augmented with three gates: Input Gate, Forget Gate and Output Gate. Thus, updated equations for the LSTMs are given in Equations 3.8 - 3.13.

Forget Gate

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{3.8}$$

Output Gate

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{3.9}$$

Input Gate

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{3.10}$$

Memory Cell

$$u_t = tanh(W_{xu}x_t + W_{hu}h_{t-1} + b_u) \tag{3.11}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{3.12}$$

Hidden state

$$h_t = o_t \odot tanh(c_t) \tag{3.13}$$

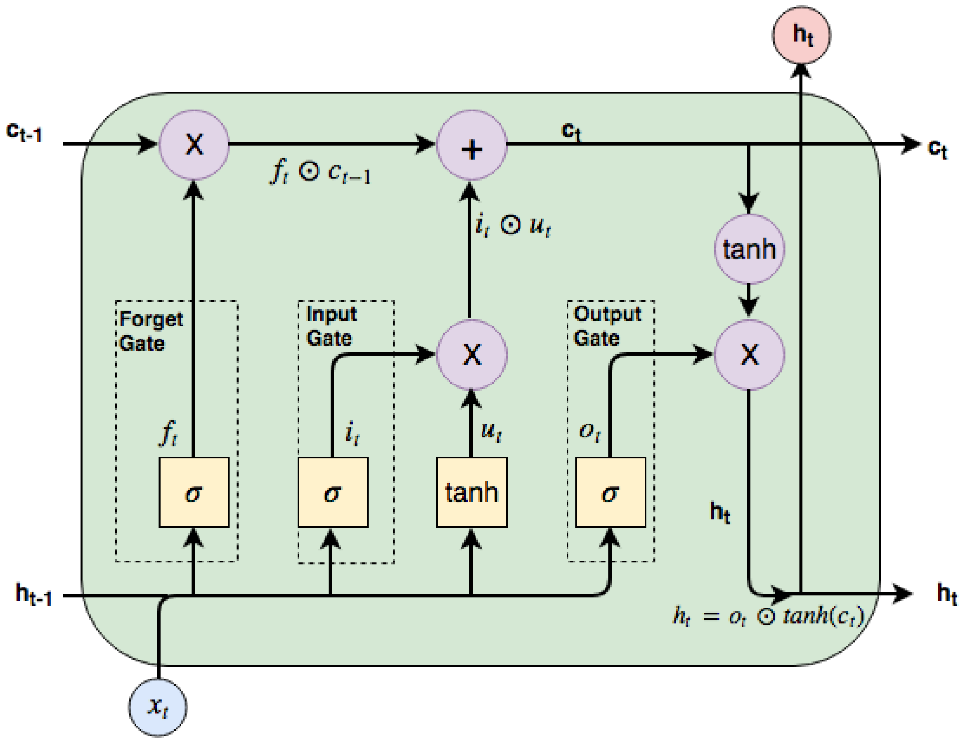


Figure 3.5: Long Short Term Memory (LSTMs) Networks with gating mechanism. Inspired from (Olah, 2015).

Gated Recurrent Unit (GRU) introduced by (Cho et al., 2014) merges the input and forget units into an ‘update’ gate while also merging the cell state and hidden state. This simplified architecture, based upon the gating mechanism of LSTMs, has increasingly been used since it reduces computational time while achieving almost similar results in many tasks. As we see later in Section 6.1, GRU cells in our generation task outperform LSTMs for the same architecture.

3.2.4 Residual Networks

As predominantly observed in Deep Learning, we can stack multiple structural units (in our case RNN units) getting a deeper network which can learn more complex data.

$$\begin{aligned} h_{1,t} &= RNN_1(h_{1,t-1}, x_t) \\ h_{n,t} &= RNN_n(h_{n,t-1}, h_{n-1,t}) \end{aligned} \tag{3.14}$$

where $h_{n,t}$ is the hidden state at layer n and time step t and $RNN(.)$ is the abbreviation for the RNN equation as described earlier (could be a complex Bi-directional LSTM). This stacking of layers can make the model learn more abstract features progressively as most commonly explained in Computer Vision literature. But this comes at an additional cost of vanishing gradient problem in vertical direction (across layers) as explained earlier for Vanilla RNN in the horizontal direction (in a single layer). This means that the layers close to the input layer will remain untrained. Residual networks (He et al., 2016) have been proposed (with the same intention as in LSTM) as a plausible solution where we directly add the output from the previous layers to the next layers (in vertical direction).

$$\begin{aligned} h_{1,t} &= RNN_1(h_{n,t-1}, x_t) + x_t \\ h_{n,t} &= RNN_n(h_{n,t-1}, h_{n-1,t}) + h_{n-1,t} \end{aligned} \tag{3.15}$$

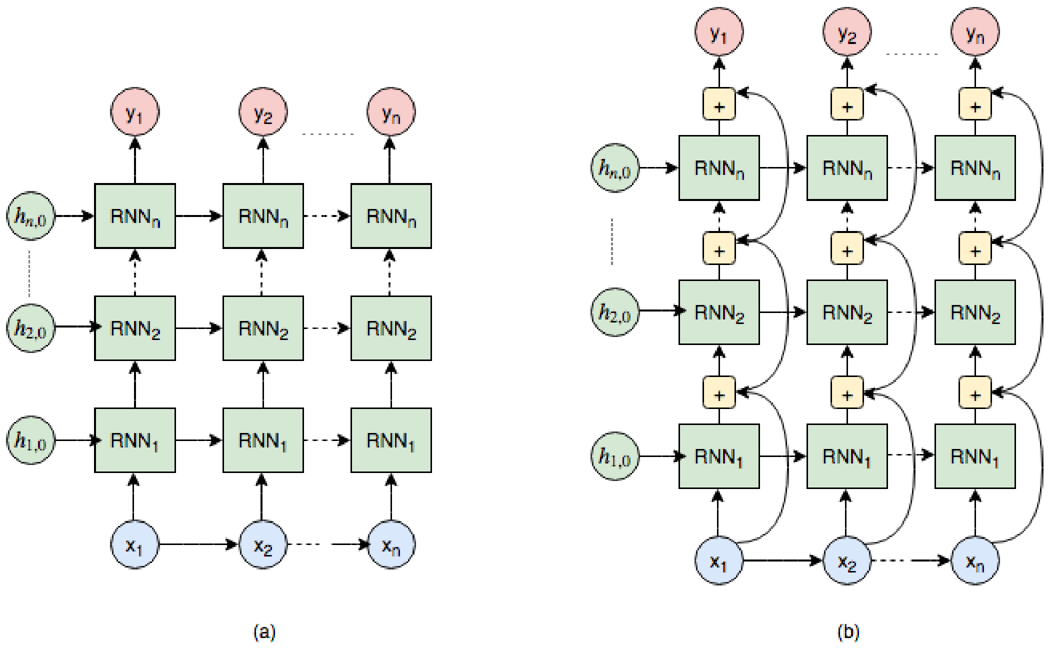


Figure 3.6: (a) Deep RNNs (b) Residual connections in Deep RNNs (Deep Residual Networks).

3.3 Sequence to Sequence Models

Sequence to Sequence models have gained prominence for NLP tasks such as Machine Translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014). Coupled with attention mechanisms, which allow the decoder to jointly align and translate, these models are used extensively to transform a given input sequence into an output sequence. We start by introducing this framework which differs during training and inference phases.

3.3.1 Encoder Decoder RNNs

Encoder Decoder RNNs, also referred as Sequence to Sequence models conceptually consist of a pipeline of two modular architectures: Encoder which *encodes* the input sequence into a 'thought vector' (also known as 'context vector' or 'summary vector') and a Decoder which generates the output sequence. Neural network approaches to NLP related tasks generally have RNN related variants (such as Bi-directional LSTMs) for both encoders and decoders while some of the recent research also include Convolutional Neural Networks (CNNs) as encoder (eg. Image Captioning (Vinyals et al., 2015; Karpathy and Fei-Fei, 2015)).

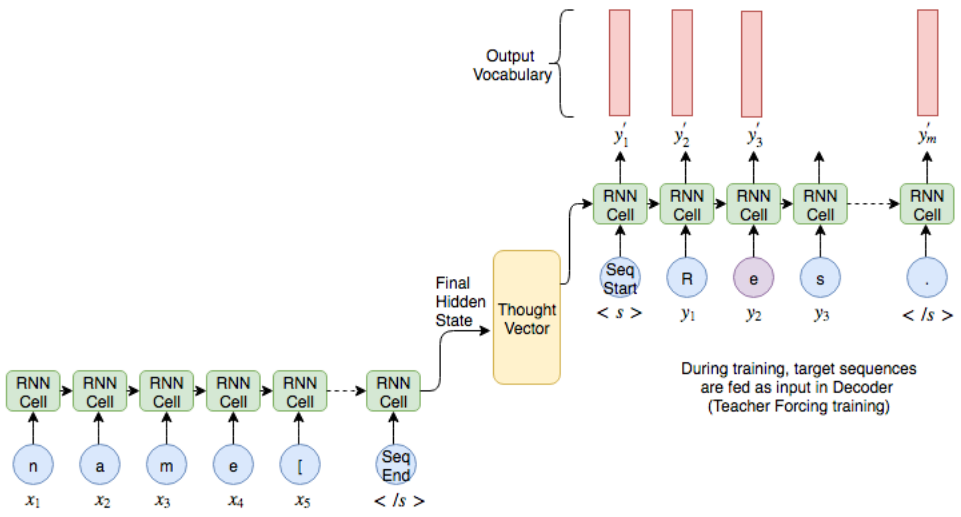


Figure 3.7: Encoder Decoder model during the training phase. Note that we provide correct target sequence at each time step in decoder which may be different from the output at previous time step, depicted in purple

Typically, we append a special start and end symbol to the sequences to take into account their variable length. An encoder takes as input a sequence of source tokens $x = (x_1, x_2, \dots, x_t, \dots, x_T)$ and produces a thought (context) vector representation c (whenever it first encounters the end symbol). Decoder takes this final hidden state of the encoder as its initial state to generate the output sequence $y = (y_1, y_2, \dots, y_t, \dots, y_{T'})$. In contrast to the RNNs described in Section 3.2, both the hidden state as well as the output at each step in the *decoder* are conditioned on the output at previous step as well as the context vector (c).

$$s_t = f(s_{t-1}, y_{t-1}, c) \tag{3.16}$$

$$P(y_t | y_{t-1}, y_{t-2}, \dots, y_1, c) = g(s_t, y_{t-1}, c) \tag{3.17}$$

where f, g represents non-linearity (such as tanh) and s_t represents the hidden state of the decoder RNN at time step t.

An important observation is that we can have input and output sequences of different lengths (T and T'). These models are generally trained by the 'Teacher Forcing' algorithm, whereby even if the decoder predicts a wrong output at a previous time step, we feed the actual target value as the input at the next time step.

To train these models, loss (typically, cross entropy) is accumulated across all the steps. The whole network can be trained jointly by backpropagation to maximize the conditional log-likelihood

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} logP(y_n|x_n,\theta) \tag{3.18}$$

where N is the number of sequences.

3.3.2 Inference

During the training stage, the input to the decoder at each step is the real target realization that we want to model. At inference, however we don't have the target sequence. The output at previous time step is provided as input at the next step to render a probability distribution over the vocabulary for the next possible symbol. Inference stops when we sample the special end token as the output. We have different ways to sample the possible output (from softmax) at each step, described further in this section.

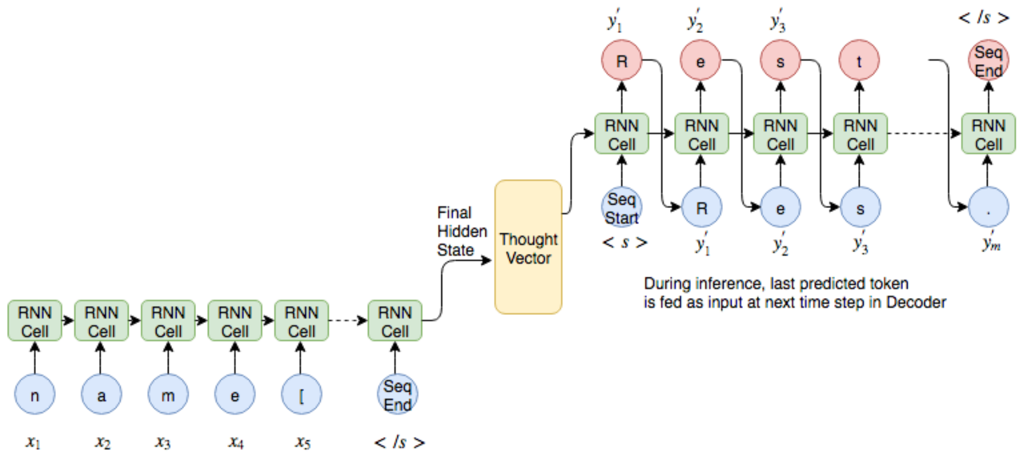


Figure 3.8: During inference, output generated at previous time step is presented as input at the next time step.

Greedy Decoding

As described earlier, during inference the output at each time step is conditionally dependent upon the previously generated output. At each time step, given the probability distribution over the possible tokens in vocabulary, we choose the one with highest probability.

$$\hat{y}_t = \operatorname*{argmax}_{y_t} P(y_t|y_{t-1},y_{t-2},\ldots,y_1,c) \tag{3.19}$$

Temperature Sampling

When using greedy search for inference, the output utterances tend to be generic and similar in nature. It is a common practice in generative algorithms to sample from the probability distribution instead of taking the maximum value from the softmax.

$$\hat{y}_t \sim P(y_t | y_{t-1}, y_{t-2}, \ldots, y_1, c) \tag{3.20}$$

We can even use a *temperature* parameter to increase the probability of the most likely token before sampling. More commonly referred as 'Temperature sampling', it works by transforming the probability distribution before sampling. Thus, output probability of each token is transformed by a 'freezing function' f shown as:

$$f_\beta(p_i) = \frac{p_i^\beta}{\sum_j p_j^\beta} \tag{3.21}$$

where Beta (β) is inverse of temperature (T). When $T \to 0$ or equivalently when β is high, it is similar to taking *argmax* as in Greedy Search.

Beam Search

Greedy decoding using *argmax* would not necessarily give the sentence with highest probability. This is due to the sequential nature of outputs where we maximize the conditional probability to generate the next probable token. Based on a dynamic programming algorithm, the beam search algorithm drastically reduces the search space from exponential size to polynomial size. Beam search in Sequence to Sequence models was introduced by (Graves, 2012; Sutskever et al., 2014) and have been used to produce top k predictions. At each time step, we retain the top k proposals (equal to *beam width*) and continue the decoding with each of them.

Length Penalty

While many strategies have been proposed to improve results using Beam Search in Machine Translation (Freitag and Al-Onaizan, 2017), we used the length normalization (aka length penalty) approach by (Wu et al., 2016) for language generation in dialogues. Neural network based approaches tend to favour shorter sentences because of the conditional modelling. When we add a new token (character) to our sequence, the overall probability of the whole sequence decreases since a negative log probability is added at each step. A heuristically derived length penalty term is added to the scoring function which ranks the probable candidates used to generate the best prediction.

$$lp(Y) = \frac{(5 + |Y|)^\alpha}{(5+1)^\alpha} \tag{3.22}$$

$$s(Y,X) = P(Y|X)/lp(Y) \tag{3.23}$$

where $\alpha \in (0, 1]$ is the length penalty factor. A value of $\alpha = 0$ reverts back to traditional beam search.

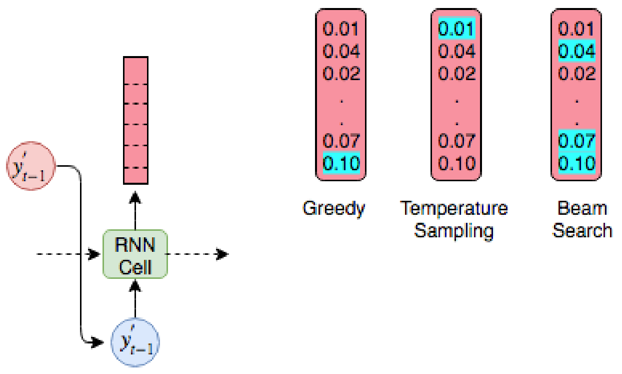
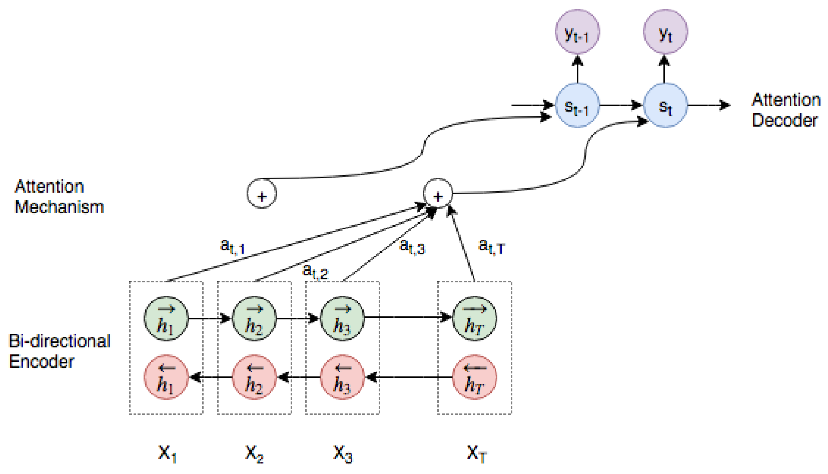


Figure 3.9: Different ways of inference at each time step. Greedy search always take the argmax from softmax probabilities while sampling based techniques 'sample' from the distribution which can be different from the maximum value. In contrast, Beam search retains top-k proposals at each step. (In the figure beam width is 3).

3.3.3 Attention Mechanism

(Bahdanau et al., 2014) introduced the attention mechanism in sequence to sequence models for Machine Translation. Described for Bi-directional encoders, the attention mechanism allows the decoder to jointly learn to align and translate, 'attending' to different parts of the source sequence. In a basic seq2seq model without attention, the onus of summarizing the whole input sequence falls onto the fixed length context vector c. As the input sequence becomes longer, it gets difficult for the encoder to include all the information in its last hidden state. The attention mechanism allows the decoder to selectively extract the relevant information at each time step.



y_{t-1}
y_t
s_{t-1}
s_t
Attention Decoder
Attention Mechanism
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
Bi-directional Encoder
$\overrightarrow{h_1}$ $\overrightarrow{h_2}$ $\overrightarrow{h_3}$ $\overrightarrow{h_T}$
$\overleftarrow{h_1}$ $\overleftarrow{h_2}$ $\overleftarrow{h_3}$ $\overleftarrow{h_T}$
X_1 X_2 X_3 X_T

Figure 3.10: Attention mechanism in Sequence to Sequence model. Here we have used Bi-directional RNN (LSTM) as the encoder. Recently, some authors have proposed using Uni-directional RNN to encode the input sequence, as it is computationally faster. They generally tend to reverse the sequence for this as suggested in the original Sequence to Sequence framework by (Sutskever et al., 2014) which though doesn't use attention mechanism.

We suppose that the forward RNN $\overrightarrow{f}$ computes the hidden states as $(\overrightarrow{h_1} \dots \overrightarrow{h_{T_x}})$ while the backward hidden states produced by backward RNN $\overleftarrow{f}$ are represented as $(\overleftarrow{h_1} \dots \overleftarrow{h_{T_x}})$. By concatenating the forward and backward hidden state for input token x_j at each time step, we can obtain corresponding 'annotation' vectors $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. Using a bi-directional RNN allows the encoder to contain the sequential dependency at each time step with the corresponding neighbours in both directions.
An 'Alignment model' is introduced as a mechanism to learn the relevancy of input around position j with the output at position i.

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.24}$$

where s_{i-1} is the hidden state of decoder at time step $t-1$ and h_j is the 'annotation' vector at the encoder level. (Bahdanau et al., 2014) used a as a feedforward Neural Network which is also jointly learnt during backpropagation. These are used to calculate the weight α_{ij} for each annotation h_j

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{3.25}$$

The *context* vector c_i is thus calculated by weighted sum of hidden annotations h_j as

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{3.26}$$

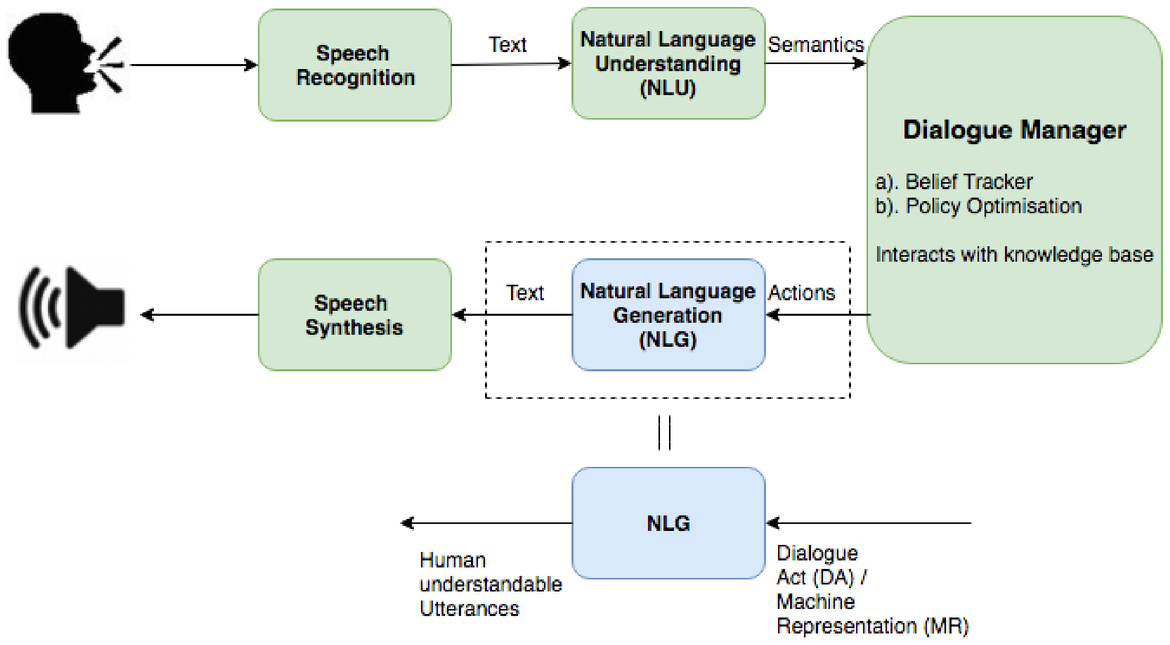
Equations 3.16 and 3.17 now take into account the context vector at each time step.

$$P(y_i | y_1, \dots y_{i-1}, x) = g(y_{i-1}, s_i, c_i) \tag{3.27}$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{3.28}$$

3.4 Spoken Dialogue Systems

Recently, there has been a lot of interest towards Spoken Dialogue Systems (SDS) in the research community. Fig 3.11 shows a schematic diagram of a Spoken Dialogue System. In addition to text based dialogue systems, it has additional modules of speech recognition and speech synthesis. Traditionally, each of the components in a Dialogue System relied upon the rule based systems. With the recent advancements in Machine Learning (Deep Learning and Reinforcement Learning in particular), there has been a shift of focus towards using Neural Networks. Research using end to end networks from the Speech Recognition to Speech Synthesis module is still at a dormant stage. Our work focuses on the Natural Language Generation (NLG) component of the Dialogue Systems using Deep Recurrent Neural Networks (Sequence to Sequence models).



Speech Recognition
Text
Natural Language Understanding (NLU)
Semantics
Dialogue Manager
a). Belief Tracker
b). Policy Optimisation
Interacts with knowledge base
Speech Synthesis
Text
Natural Language Generation (NLG)
Actions
NLG
Human understandable Utterances
Dialogue Act (DA) / Machine Representation (MR)

Figure 3.11: Systematic components of a Spoken Dialogue System

3.5 NLG for Spoken Dialogue Systems

Natural Language Generation (NLG) plays a central role in Spoken Dialogue Systems, whereby we convert MRs (Meaning Representation) or Dialogue acts (DA) into human understandable meaningful utterances. A typical excerpt of this kind of pair taken from the dataset that we describe later in Section 4.1:

MR: name[The Eagle], eatType[coffee shop], food[French], priceRange[moderate], customer-Rating[3/5], area[riverside], kidsFriendly[yes], near[Burger King]
RF: The three star coffee shop, The Eagle, gives families a mid-priced dining experience featuring a variety of wines and cheeses. Find The Eagle near Burger King.

Figure 3.12: An example (MR,RF) pair in E2E NLG Challenge dataset.

The Neural Network (NN) based approach to NLG, recently introduced by (Wen et al., 2015), works at the word level and require both pre- as well as post-processing. They resort to a process called *de-lexicalisation* whereby each slot value is replaced by special placeholders before training. Our work by contrast is inspired by (Goyal et al., 2016) who also used character based seq2seq for dialogue generation.

— 4 —
Dataset, Model and Approach

In this chapter, we will first discuss the statistics and our approach towards tackling the E2E NLG Challenge. When we directly try to generate the predictions from the Meaning Representations (MRs), we term this as forward predictions. In our further discussions, we have *forward* models trying to predict the output utterances and generating top-k probable candidates (n-best list) for each MR. In addition to this, we have *reverse* models which are used as a mechanism to score and re-rank this n-best list to produce the best utterance for each MR. This chapter is further divided into Forward models and our proposed re-ranking methods.

4.1 E2E NLG Challenge

Our work is motivated by a future submission to the recently released End-2-End Natural Language Generation Challenge. This dataset comprises of 50K (MR,RF) pairs, MR being the Meaning Representation and human ReFerence as RF. The slot-value pairs are also canonically ordered and systematically structured in the Meaning Representation. Table 4.1 shows the statistics of *slot type*, categorized by the number of slots in each MR.

(Novikova et al., 2016) explain the protocol followed for crowdsourcing *E2E NLG Challenge* dataset. Slightly different from the description in the article, there are two additional slots: 'kidsFriendly' and 'children-friendly' which seem to be alternates for 'familyFriendly' [1].

Thus, there are in total 10 slots (in the order in which they appear in Meaning Representation) described by their slot values below:

- **name:** 34 unique slot values which are *not* in a uniform frequency distribution
- **eatType:** 3 unique values - 'coffeee' (60%) , 'pub' (27%), 'restaurant' (13%)

[1] At the time of writing of this thesis, challenge organizers released an updated version of the dataset incorporating our earlier comments to them, which we describe as follows. The original version of E2E NLG Challenge data contained a few erroneous newline characters (Line 603 in devset.csv as well as 30048 in trainset.csv). There were different character encodings for MR and RF, which we uniformized to utf-8. Also, there were a few wrongly encoded characters (such as on line 23191 in trainset.csv, apart from two non-ascii characters: £ and é) as well a minor spelling error of slot value 'coffee' as 'coffeee'. In contrast to previous version, the new version of the dataset doesn't contain the additional slots 'kidsFriendly' and 'children-friendly'. As per our findings, these three slots were used interchangeably by the crowd sourcers when annotating the data. Thus, the recent version now contains only 8 slots with the other two slots ('kidsFriendly' and 'children-friendly') mapped to 'familyFriendly'.

Num Slots/ SlotType	**area**	**children-friendly**	**customer rating**	**eat Type**	**family Friendly**	**food**	**kids Friendly**	**name**	**near**	**price Range**	**Total**
3	1%	0.3%	2%	1%	1%	3%	1%	5%	1%	2%	5%
4	6%	2%	9%	5%	3%	13%	2%	18%	5%	9%	18%
5	15%	5%	18%	12%	6%	23%	5%	29%	12%	19%	29%
6	21%	6%	22%	17%	9%	25%	6%	29%	17%	22%	29%
7	13%	4%	13%	12%	5%	15%	4%	15%	12%	14%	15%
8	4%	1%	4%	4%	1%	4%	1%	4%	4%	4%	4%
Total	60%	19%	68%	51%	25%	83%	19%	100%	50%	68%	100%

Table 4.1: This table shows the frequency distribution of different slot types categorized by the number of slots in the Meaning Representation. 'name' slot is always present in the MR

- **food:** 7 unique values ('Chinese', 'English', 'Fast Food', 'French', 'Indian', 'Italian', 'Japanese') in an almost uniform ratio
- **priceRange:** 6 unique values (uniform ratio)
- **customerRating:** 6 unique values (uniform ratio)
- **area:** 'city centre' (32%), 'riverside' (68%)
- **familyFriendly:** no (42%), yes (58%)
- **children-friendly:** no (25%), yes (75%)
- **kidsFriendly:** no (28%), yes (72%)
- **near:** 19 unique slot values which are *not* in a uniform frequency distribution

4.2 Forward Models

The RNN based approach, introduced by (Wen et al., 2015) for dialogue acts, generates the output word-by-word, involving both pre-and post-processing of the inputs and outputs respectively. They resort to special delexicalisation process whereby named entities (slot values) are replaced by corresponding placeholders before training. After inference, these entities are transformed back (re-lexicalised) corresponding to the original value in the MR. This approach is used to resolve the rare or unknown word problem. As discussed in (Goyal et al., 2016), this de-lexicalization loses any morphological information (eg. gender-specific information) present in the slot value. Additionally, we also have to de-lexicalise the multiple slots of same type using sub-categories (though not relevant in our E2E NLG Challenge).

[MR]: name[<NAME>], eatType[<EATTYPE>], near[<NEAR>]
[RF]: <NAME> is a <EATTYPE> near <NEAR>.

Figure 4.1: De-lexicalisation procedure.

The motivation to use char2char models for dialogue generation comes from the above mentioned drawbacks of the models where we do not have to employ any pre- or post-processing (not even tokenization or lowercasing) technique. We used Character based Sequence to Sequence with attention mechanism, whereby the input to the encoder is the character embeddings of the source (MR) Meaning Representation and the output is also generated char-by-char; this approach avoids the rare word problem, as the character vocabulary is very small. This kind of model is trained using 'Teacher Forcing' algorithm. Figure 4.2, borrowed from (Britz et al., 2017), provides an overview of the framework.

Probabilistically, we want to find a prediction y that maximizes the probability of y conditioned on the source dialogue act, i.e. $\text{argmax}_y P(y|x)$. Here the input $x = (x_1, x_2, \ldots x_n)$ and the output $y = (y_1, y_2, \ldots y_m)$ both are the sequence of characters.

Compared to (Goyal et al., 2016), our forward model is *deeper* with more number of layers in both encoder and decoder RNN. While they used greedy search and beam search (of beam width 5) for inference, we found that Temperature sampling outperforms the greedy search as well as traditional Beam search with 0 length penalty. However, using length penalty equal to 1 provides the best solutions in terms of BLEU. As we will see later in Section 6.1, the forward model already works quite well for certain aspects (apart from some cases of omissions). These forward predictions form our baseline and we further explored re-ranking techniques to improve decoding results.

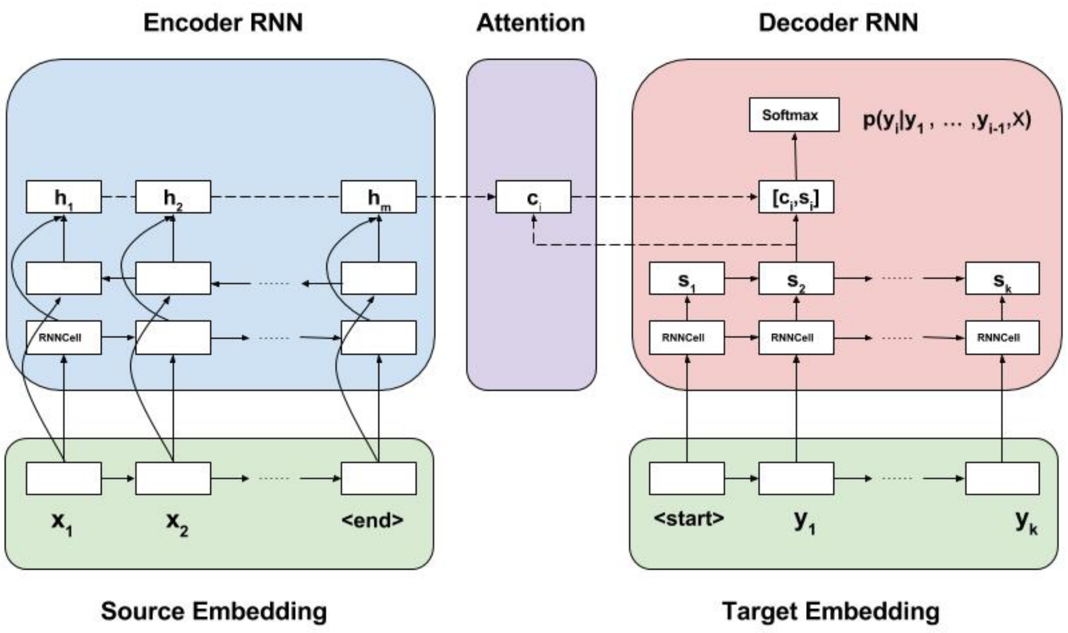


Figure 4.2: Schematic diagram of attention based seq2seq architecture. Contrary to word-based sequences, we use character-based sequences for generating grammatically correct and natural utterances. Figure borrowed from (Britz et al., 2017)

4.3 Re-ranking

During decoding, we use approximation techniques such as Greedy, Sampling or Beam Search. An additional benefit of using Beam Search, apart from increasing the search space at each time step, is that it allows us to produce n-best list (top-k probable candidates). Thus, it makes sense to re-rank these predictions, producing the best generated utterance. In the following subsections, we discuss two of the strategies that we propose for re-ranking.

4.3.1 Reverse Models

While we train *forward* models to generate probable utterances from the input MR, we can also train *reverse* models to do exactly the opposite, i.e generate inverse reconstructions of the MR for the output probable predictions (See Fig. 4.3). The idea behind this methodology is that if there is any kind of omission (or any other error such as addition of material) in the prediction, the reconstructed MR for this prediction should also tend to omit stuff.

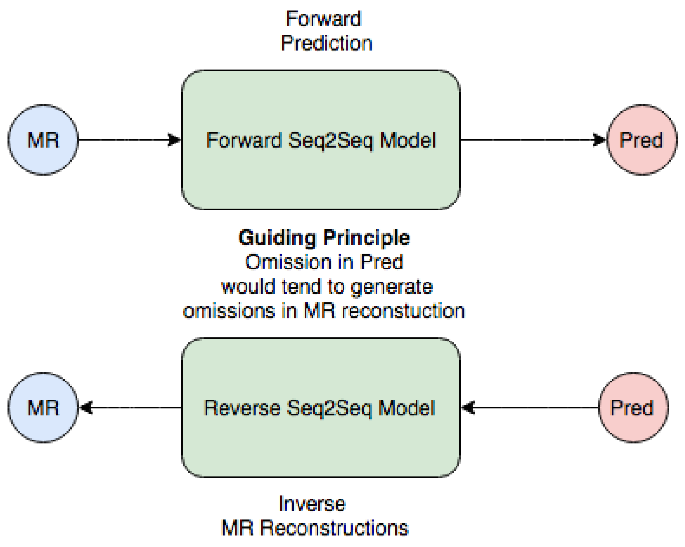


Figure 4.3: In the forward direction we try to model RF using MR. In the reverse direction we construct inverse generations of the MR from RF.

Edit distance or Levenshtein distance (Navarro, 2001) is commonly used in NLP to determine dissimilarity between two strings. It works by counting the minimum number of operations required to transform one string into the other. Traditionally, re-ranking approaches (commonly used in Information Retrieval (IR)), require combining the scores using some kind of weighted parameters. To avoid defining these weights, we used a more simplistic and natural mechanism for the same. At the time of re-ranking, we choose the first output in our n-best list with a 0 edit distance. If no such prediction can be found, we rely upon the first prediction in our probabilistically sorted n-best list. The pipeline for our approach is depicted in Fig. 4.4.

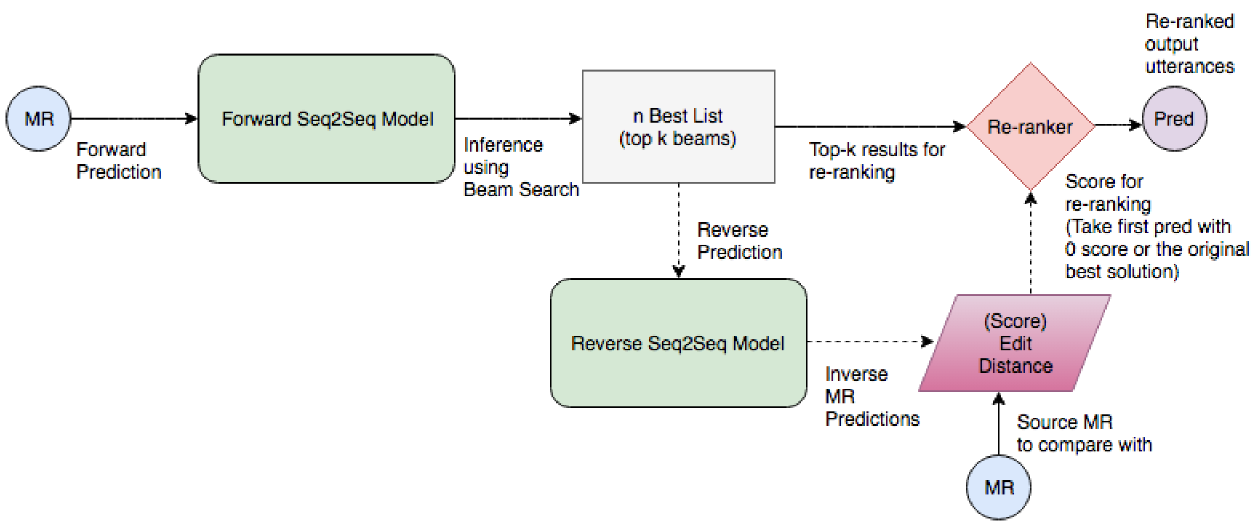

Re-ranked output utterances
MR
Forward Prediction
Forward Seq2Seq Model
Inference using Beam Search
n Best List (top k beams)
Top-k results for re-ranking
Re-ranker
Pred
Score for re-ranking (Take first pred with 0 score or the original best solution)
Reverse Prediction
Reverse Seq2Seq Model
Inverse MR Predictions
(Score) Edit Distance
Source MR to compare with
MR

Figure 4.4: Illustration of the pipeline for re-ranking approach. Apart from Forward and Reverse seq2seq models, we have a re-ranker based on the edit distance of the actual MR and the inverse reconstructed MR.

4.3.2 Error detection by classification

As an alternate approach for detecting errors in our generated utterances, we apply a 'bi-sequence' classification (labelling) problem. Thus, given a MR and its prediction by the Forward model, we want our classifier to output a label of 1 if there is no omission or a value of 0 if omission has been detected by the classifier. We use the 'Siamese' architecture described in (Mueller and Thyagarajan, 2016; Neculoiu et al., 2016) for this purpose.

Protocol for creating dataset

We artificially create a training set for the classifier to detect errors (primarily omission) in the generated utterances by a data augmentation technique. As far as we know, this is the first approach of using data augmentation in this way for the NLG task. Because of the systematic structure of the slots in MR, it gives us freedom to naturally augment data for our use case. This kind of data augmentation procedure, though very natural, opens up great directions to create artificial datasets for Natural Language Generation. We define the procedure first for creating a dataset to detect omission and then we show how a similar approach can be used to create augmented data to detect additions.

The basic idea assumes that there are no omissions in the RF for a given MR. These can be considered as positive pairs when detecting omissions. Now if we artificially add another slot to the original MR and use the same RF for this new MR, naturally the RF tends to show omission of the added slot.

$$MR_{original} \xrightarrow{+\text{ Missing slot}} MR_{new} \tag{4.1}$$

This is now a two stage procedure:

- Select a slot to add.
- Select a corresponding slot value.

Instead of sampling a particular slot to add, we add all the slots that could be augmented in the MR apart from the currently present slots, one by one. We explain the procedure using a MR example from the dataset.

MR: name[The Punter], customer rating[high], area[riverside]

For notations in further discussion, we define *friendlyTuple* as ('kidsFriendly', 'children-friendly', 'familyFriendly') and *presentTuple* as the tuple of slot types present in the MR (for which we augment data). We have *allSlotsTuple* as a tuple of all the 10 possible slots described earlier. We also define *missingTuple*, which contains the slots not present in MR from *allSlotsTuple*. Thus, for the given example above ,we have:

- **allSlotsTuple**: (name, eatType, food, priceRange, customerRating, area, familyFriendly, children-friendly, kidsFriendly, near)
- **presentTuple**: (name, customerRating, area)
- **friendlyTuple**: (familyFriendly, children-friendly, kidsFriendly)
- **missingTuple**: (eatType, food, priceRange, familyFriendly, children-friendly, kidsFriendly, near)

All the slots in *friendlyTuple* are used exclusively and thus if any one of the elements is present in the MRs, other slots of that kind would not be present in the MR. We thus maintained this restriction by using the procedure described below.

$$\text{missingTuple} = \begin{cases} \text{allSlotsTuple} \setminus (\text{presentTuple} \cup \text{friendlyTuple}) & \text{if}(\text{presentTuple} \cap \text{friendlyTuple}) \neq \emptyset \\ \text{allSlotsTuple} \setminus \text{presentTuple} & otherwise \end{cases}$$

To understand this more clearly, we have the number of elements in this presentTuple as NumPres while the length of allSlotsTuple and friendlyTuple is 10 and 3 respectively. NumMiss denote the number of slots in missingTuple.

$$\text{NumMiss} = \begin{cases} 10 - \text{NumPres} - 2 & \text{if}(\text{presentTuple} \cap \text{friendlyTuple}) \neq \emptyset \\ 10 - \text{NumPres} & otherwise \end{cases}$$

Having this *missingTuple*, for each element in the set, we create new MRs by adding just one element creating the number of MRs equal to NumMiss. For all of these new MRs, we use the same $RF_{original}$ as for the $MR_{original}$. To avoid class imbalance, we also replicate the $MR_{original}$ by the same number (NumMiss) in our new dataset.

Thus, we have triplets of the form *originalTriplet* ($MR_{original}$,$RF_{original}$,1) and *constructedTriplet* (MR_{new},$RF_{original}$,0). By construction, we have same number of both kind of triplets. Here label 1 is to define no omission and 0 for a case of omission. Each row of *constructedTriplet* type would depict omission as the $RF_{original}$ was crowd sourced based on $MR_{original}$ and thus naturally omits the new slot we artificially added to form MR_{new}. Having chosen the slot type to be added, we add the slot value according to the probability distribution of the slot values for that slot type defined in Section 4.1.

On the other hand, when we want to create dataset which would be used for training our model to detect additions, we systematically remove one slot in the original MR to create new MRs. We did not remove 'name' slot as it was present in all the MRs.

$$MR_{original} \xrightarrow{\text{- Selected slot}} MR_{new} \tag{4.2}$$

In both cases, we could control the procedure by manipulating MR instead of the Natural Language RF for detecting both additions and omissions. This kind of augmented dataset (having triplets) opens up the possibility of using Siamese kind networks to detect these kinds of errors.

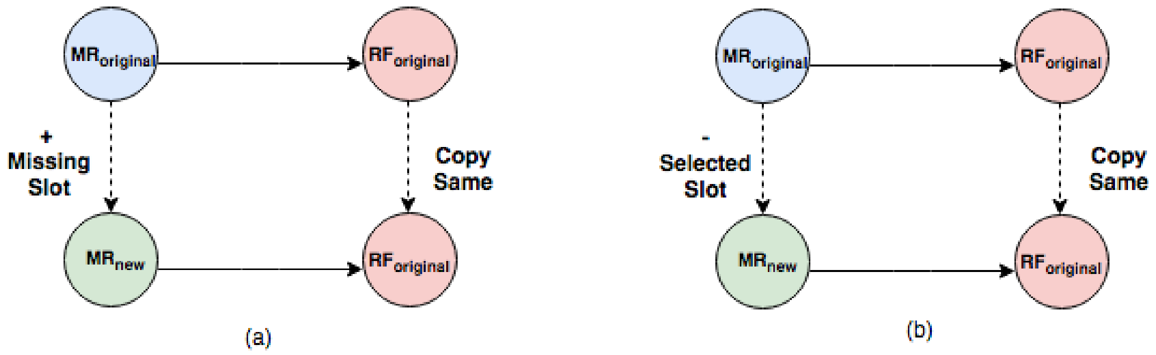


Figure 4.5: This figure shows the our approach towards creating augmented datasets. (a) shows the procedure to create a dataset which can be used to train our model to detect omissions. (b) on the other hand creates a dataset to detect additions.

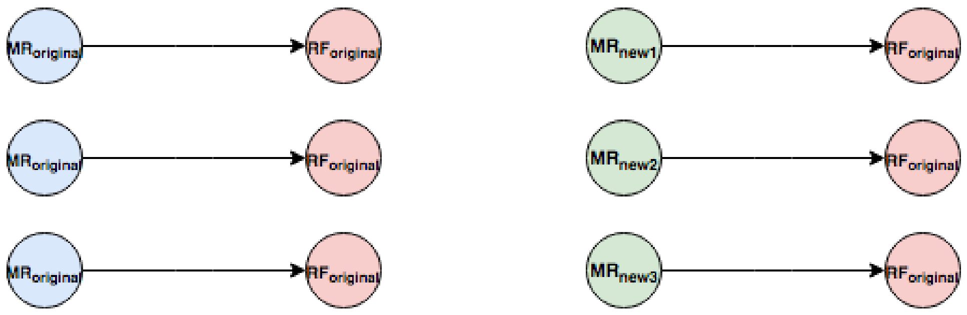


Figure 4.6: We replicate the original (MR,RF) pair by the same number of artificially generated new pairs of fabricated (MR,RF).

Dataset for Classifier

We artificially created the dataset to train the classifier using the protocol defined above. Thus, for each (MR, RF) pair, we created different number of new fabricated MRs dependent upon the number of missing slots in the original MR. One point to note is that the three slots ‘kidsFriendly’, ‘children-friendly’ and ‘familyFriendly’ are used interchangeably in the dataset which affects our data generation. Here, we show an example for the MR with total 4 slots (including ‘kidsFriendly’ slot) for simplicity.

MR: name[The Punter], customer rating[high], area[riverside], kidsFriendly[yes]
RF: The Punter is a kid friendly restaurant in the riverside area with a high customer rating.

Thus, new produced MRs for this example in E2E NLG dataset are:

- name[The Punter], eatType[coffeee shop], customer rating[high], area[riverside], kidsFriendly[yes]
- name[The Punter], food[Indian], customer rating[high], area[riverside], kidsFriendly[yes]
- name[The Punter], priceRange[more than 30], customer rating[high], area[riverside], kidsFriendly[yes]
- name[The Punter], customer rating[high], area[riverside], kidsFriendly[yes], near[Cafe Rouge]

For all these produced MRs, RF would be the same, thus producing 4 triplets of the form (MR_{new},$RF_{original}$,0). Simultaneously, we will also add 4 triplets of the form ($MR_{original}$,$RF_{original}$,1) in our dataset to maintain class balance.

Similarly, to detect addition of material in our predictions, we create an artificial dataset whereby we artificially remove slot-value pair from the original MR. The new MRs thus to detect addition are:

- name[The Punter], area[riverside], kidsFriendly[yes]
- name[The Punter], customer rating[high], kidsFriendly[yes]
- name[The Punter], customer rating[high], area[riverside]

A key observation to be made is that we maintain the order of slot value pair as was observed in our original dataset.

Description of Classifier (Scorer)

Our aim, now, is to construct a classifier which could identify any omission (can be used in a similar fashion for detecting addition) given a pair of (source, target) elements (MR, Pred). Having created the dataset described in previous section, we will train this model to learn the representation of source and target sequences and to identify if there is any omission by calculating the dissimilarity between these representations.

The model we construct is based upon the Recurrent Siamese architectures described in (Mueller and Thyagarajan, 2016; Neculoiu et al., 2016) which were used to calculate sentence similarity. In contrast to these approaches as well as the traditional Siamese Networks, there is no weight sharing between the two branches of our network while we have a common embedding layer at the input. We use Bi-directional LSTMs for generating representations at both ends of the network. (See Fig 4.7). We denote them by *a* and *b*. Thus, we have y=1 if *a* and *b* are 'similar' (no omission) and 0 otherwise

We are further exploring ways to define this distance *d* and the corresponding loss. For training, we need a differentiable loss based on d. Following (Mueller and Thyagarajan, 2016), we are exploring L1 distance (taking values between 0 and ∞) as the distance, and the following loss value:

$$L = (y - exp(-d(a,b)))^2 \tag{4.3}$$

In inference, we would then choose the label y = 0 or 1 which would lead to a smaller loss. For y=1, the loss is large when the distance is small. For y=0, loss is small when the distance is large. Here for $exp(-d(a,b)) > 0.5$, i.e when $d(a,b) < -log(0.5)$, we choose 1 and 0 otherwise. Another distance-loss pair we can try in case this strategy does not work:

$$L = (y - (1 - d(a,b)))^2 \tag{4.4}$$

where d can be any distance metric between 0 and 1 (like cosine distance).

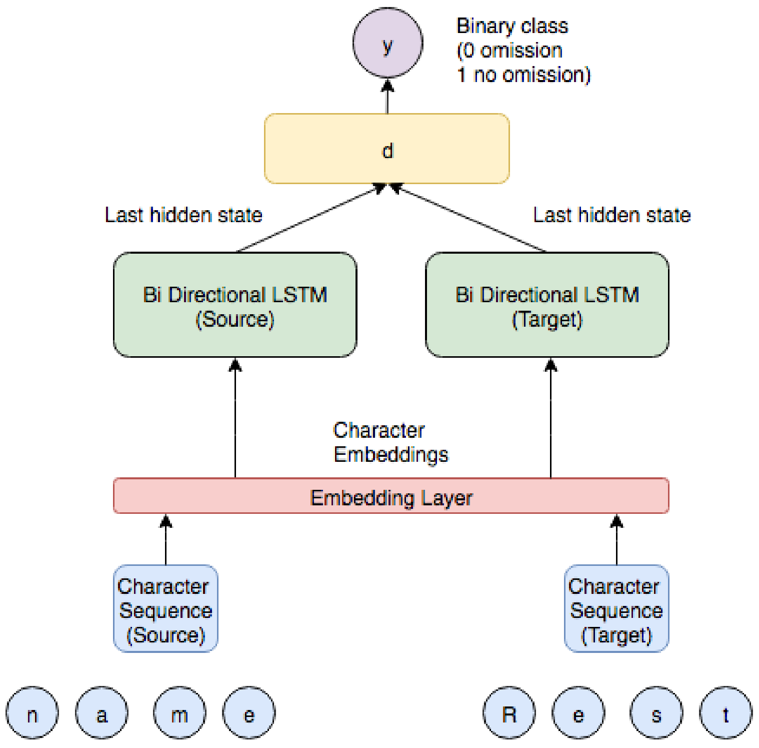


Figure 4.7: Proposed Siamese kind model for detecting omissions

— 5 —
Implementation and Metrics

5.1 Implementation

Recently, Google Brain released its open source *tf-seq2seq* framework[1], built over TensorFlow (Abadi et al., 2016) in Python, and provided along with (Britz et al., 2017), with some standard configuration options. Though targeting models for word level Machine Translations, it is released as a general purpose encoder-decoder framework for different tasks using seq2seq models. However, to directly use this framework for our character based models, we had to replace non-ascii characters (such as £ and é) with ascii characters not present in our source and target vocabulary ($ and ^ symbol respectively) as we faced file encoding related issues [2] We initially also used the ordinal integer value of the characters in utf-8 encoding to transform the sequence (and handle these special characters) but finally switched to the former solution for evaluating the model at the intermediary steps while training.

This tf-seq2seq toolkit trains on pairs of sequences presented in parallel text format (separate source and target sequence files). While many options are configurable (number of layers, unidirectional *vs* bidirectional encoder, additive *vs* multiplicative attention mechanism, GRU Cho et al. (2014) *vs* LSTM cells Hochreiter and Schmidhuber (1997), etc.), the core architecture is common to all models. This is by now a pretty standard attention-based encoder-decoder architecture based on Bahdanau et al. (2014); Luong et al. (2015).

All other implementations (eg. to create a classifier as well as to calculate edit distance) are also done using Tensorflow and Python.

5.2 Metrics

Automatic evaluation metrics in NLG are still evolving, which can judge the generated predictions, not only for Semantic Adequacy and Linguistic quality but also from the point of

[1] `https://github.com/google/seq2seq` .

[2] These non-ascii characters are represented by two bytes in utf-8. Though the framework encodes all the text to utf-8, but tensorflow string split function (`tf.string_split`) considers special characters in utf-8 as two different characters, thus raising unicode codec error. Issue #153 `https://github.com/google/seq2seq/issues/153`. This would not be a case when we use word level models using this framework since whitespace is used as delimiter instead of an empty string.

diversity in produced utterances. Though BLEU score is quite popular as an automatic evaluation metric, researchers still rely on Human Evaluation to better understand the functioning of their models.

5.2.1 Automatic Evaluation

We used BLEU score for automatic evaluation which we describe below.

BLEU

BLEU (Papineni et al., 2002) score is a common metric used in Machine Translation and NLG literature to evaluate the performance of produced sequences. In a nutshell, BLEU calculates the n-gram overlap between the text generated by the model with respect to a human reference. BLEU, though standardly used as an automatic evaluation, its relevancy as well as the semantic information it captures is still debatable. We thus, have to resort to Human Evaluation to better judge our produced utterances.

5.2.2 Human Evaluation

The human evaluations are done by two annotators on the top 20 predictions (n-best list from beam search) of the previously discussed model, for the first 100 MRs of the devset. We define the following metrics for our evaluation:

1. **Semantic Adequacy**

   a) Omission [1/0]: information present in the MR that is omitted in the predicted utterance (1=No omission, 0=Omission).

   b) Addition [1/0]: information in the predicted utterance that is absent in the MR (1=No addition, 0=Addition).

   c) Repetition [1/0]: repeated information in the predicted utterance (1=No repetition, 0=Repetition).

2. **Linguistic Quality**

   a) Grammar [1/0]: (1=Grammatically correct, 0=incorrect). Note: one annotator punished the model even for (rare) mistakes of punctuation.

   b) Naturalness [2,1,0]: subjective score to measure the naturalness of the utterance (2 being best).

   c) Comparison to reference [1/0/-1]: subjective score comparing the prediction with the crowdsourced RF. ('vsRef' in the Table 6.2, 1=Prediction better than RF, 0=Prediction at par with RF, -1=RF better than prediction).

3. **Oracle [1/0/-1]:** We define *oracle* as a generated utterance which could be considered as perfect in both adequacy as well as linguistic quality. Thus, we gave a score 1 if the first prediction is itself an oracle, 0 when the oracle is found in the top 20, and -1 when no oracle is found there.

— 6 —
Experiments and Results

We organize this chapter into two sections. First, we discuss the quality of results produced by our forward models. Human Evaluation based on the metrics defined in the previous section, show promising direction of our proposed approach of re-ranking the top-k probable candidates.

6.1 Forward Models

Similar to (Britz et al., 2017), we fixed all the other hyper-parameters other than the one under study for our experiments. This greedy type search for hyper-parameters was exploited against grid search because of a very large search space.

Model Specification	**Beam Width**	**Length Penalty**	**Depth (Number of layers)**				
Encoder			1	1	2	4	4
Decoder			1	2	2	4	4
Cell Unit			GRU	GRU	GRU	GRU	LSTM
Greedy Search			20.94	22.59	23.5	23.84	23.98
Beam Search	Beam 5	LP0	15.85	22.47	21.76	22.73	20.15
	Beam 10	LP0	14.5	21.4	19.98	21.15	18.88
	Beam 20	LP0	13.48	20.18	18.5	19.94	17.93
	Beam 5	LP1	20.64	24.77	24.67	**24.94**	23.87
	Beam 10	LP1	21	25.05	24.88	24.69	24.27
	Beam 20	LP1	21.27	25.4	24.96	**24.6**	24.05

Table 6.1: BLEU scores on devset while varying the depth of both encoder and decoder RNNs, type of cell unit, different beam widths and length penalty. (Results reported for only a single experiment with training and prediction.)

Taking cue from the recommended configurations in Table 7 of (Britz et al., 2017) and the provided example configs in *tf-seq2seq*, we experimented with different numbers of layers in the encoder and decoder as well as different beam widths, while using the bi-directional encoder along with the 'additive' attention mechanism. We report the BLEU scores[1] for different configurations of the seq2seq model in Table 6.1. In our initial experiments, using a beam-width

[1]Calculated using the multi-bleu perl script bundled with *tf-seq2seq*.

of 5 (with no length penalty), with 4 layers in both the encoder and decoder and GRU cells, showed the best results in terms of BLEU (score of 24.94).

We observed significant improvements using a length penalty of 1, and decided to use this architecture as a basis for the human evaluations, with a beam-width of 20 to facilitate the observation of oracles. These evaluations were thus conducted on [encoder 4 layers, decoder 4 layers, GRU cell, beam-width 20, length penalty 1], though we found slightly better performing models in terms of BLEU at a later stage. Based on the metrics defined in Section 5.2.2, the human evaluations were performed by two annotators on the top 20 predictions of this model, for the first 100 MRs of the devset[2].

We show a few examples of utterances (most probable or in top position) produced by our model for further discussion.

1. **[MR]:** name[The Punter], customer rating[high], area[riverside], kidsFriendly[yes]
   [RF]: *In riverside area, there is The Punter, which is high rated by customers and kids are friendly.*
   [Pred]: The Punter is a kid friendly restaurant in the riverside area with a high customer rating.

2. **[MR]:** name[The Golden Palace], eatType[coffeee shop], food[Japanese], priceRange[£20-25], customer rating[high], area[riverside]
   [RF]: *For highly-rated Japanese food pop along to The Golden Palace coffee shop. Its located on the riverside. Expect to pay between 20-25 pounds per person.*
   [Pred]: The Golden Palace is a coffee shop providing Japanese food in the £20-25 price range. It is located in the riverside area.

3. **[MR]:** name[Strada], food[Fast food], priceRange [moderate], customer rating[1 out of 5], kidsFriendly[no], near [Rainbow Vegetarian Cafe]
   [RF]: *Strada is a Fast food restaurant near the Rainbow Vegetarian caffe which has a moderate customer rating of 1 out of 5 for a non Kids friendly restaurant*
   [Pred]: Strada is a moderately priced fast food restaurant in the **moderate price range**. It is located near Rainbow Vegetarian caffe.

Among the utterances produced by the model in first position (Pred), the most prominent issue was that of omissions (underlined in example 2). There were no additions or non-words (which was one of the primary concerns for Goyal et al. (2016)). We observed only a couple of repetitions which were actually accompanied by omission of some slot(s) in the same utterance (repetition highlighted in bold in example 3).

Surprisingly enough, we observed a similar issue of omissions in human references (target for our model). We then decided to perform comparisons against the human reference ('vsRef' in Table 6.2). Often, the predictions were found to be semantically or grammatically better than the human reference; for example observe the underlined portion of the reference in the first example. The two annotators independently found the predictions to be mostly grammatically correct as well as natural (to a slighty lesser extent).[3]

[2]These annotations are accessible at `https://docs.google.com/spreadsheets/d/1wMu42g8bzyFxBUJ33QIdkqN3md3281pg6rLGrnDbEIE/edit?usp=sharing`

[3]Annotator-1 was more severe in highlighting even the (rare) punctuation issues as grammatical mistakes. There was also a slight disagreement with Annotator-2 being more severe than Annotator-1 when assessing the references against the predictions.

A general feeling of the annotators was that the predictions, while showing a significant amount of linguistic diversity and naturalness, had a tendency to respect grammatical constraints *better* than the references; the crowdsourcers tended to strive for creativity, sometimes not supported by evidence in the MR, and often with little concern for linguistic quality; it may be conjectured that the seq2seq model, by "averaging" over many linguistically diverse and sometimes incorrect training examples, was still able to learn what amounts to a reasonable linguistic model for its predictions.

We also investigated whether we could find an 'oracle' (perfect solution as defined in section 5.2.2) in the top-20 predictions and observed that in around 70% of our examples the oracle could be found in the top results (see Table 6.3), very often (51%) in first position. In the rest 30% of the cases, even the top-20 predictions did not contain an oracle. We found that the presence of an oracle was dependent on the number of slots in the MR. When the number of slots were 7-8, the presence of an oracle in the top predictions using beam width of 20 decreased significantly to approximately 40%. In contrast, with 4 slots, our model predicted an oracle right at the first place for 83% of the cases.

Ann	**O(1/0)**	**A(1/0)**	**R(1/0)**	**G(1/0)**	**N(2/1/0)**	**vsRef(1/0/-1)**	**Or(1/0/-1)**
Ann 1	51/49	100/0	97/3	93/7	85/13/2	46/16/38	50/18/32
Ann 2	51/49	100/0	98/2	98/2	80/18/2	29/36/35	51/18/31
Mean	51/49	100/0	97.5/2.5	95.5/4.5	82.5/15.5/2	37.5/26/36.5	50.5/18/31.5

Table 6.2: Human annotations for 100 samples using different metrics defined in Sec. 5.2.2. O (Omission), A (Addition), R (Repetition) and G (Grammar) are on binary scale. Naturalness is measured as (2/1/0) and Oracle as (1,0,-1). Predictions were also judged against the reference on a scale of (1,0,-1).

Slots	**DA**	**Or@1**	**Or**	**No Or**
3	1(1%)	1(100%)	0(0%)	0(0%)
4	29(29%)	24(83%)	3(10%)	2(7%)
5	25(25%)	13(48%)	6(24%)	6(28%)
6	29(29%)	11(34%)	5(17%)	13(48%)
7	11(11%)	1(9%)	3(27%)	7(64%)
8	5(5%)	1(20%)	1(20%)	3(60%)
Total	100	51	18	31

Table 6.3: Human annotations for different slots using beam-width 20. 'Or@1' represents the presence of an 'oracle' at first position while 'Or' represents the presence of 'Oracle' (desirable) in the top-20 predictions. Cases where no oracle was found are marked as 'No Or'.

As an alternative search strategy, we implemented Temperature Sampling described in Section 3.3.2. We experimented with different values of Beta (β), which is inverse of temperature. Table 6.2 show our results for different beta values when we use temperature sampling.

For the Beam Search, we found a direct correlation of the BLEU score with the length penalty. (See Table 6.1).

Search	**Parameter**	**BLEU**
Greedy	-	23.84
B5	LP0	22.73
B10	LP0	21.15
B20	LP0	19.94
B5	LP0.5	24.1
B10	LP0.5	23.5
B20	LP0.5	23.1
B5	LP1	**24.94**
B10	LP1	24.69
B20	LP1	24.6

Table 6.4: We experimented with different length penalties and found Length Penalty 1 to give the best predictions in terms of BLEU.

Search	**Parameter**	**BLEU**
Greedy	-	22.59
B5	LP0	22.47
B10	LP0	21.4
B20	LP0	20.18
B5	LP1	24.77
B10	LP1	25.05
B20	LP1	**25.4**
Sampling	-	12.91
Sampling	Beta 10	22.63
Sampling	Beta 20	**22.84**
Sampling	Beta 40	22.67
Sampling	Beta 50	22.6
Sampling	Beta 100	22.61

Table 6.5: Comparison of temperature sampling against Greedy and Beam search. For a Beta value of 20, we obtained results, which were better than Greedy as well as Beam Search (with 0 Length Penalty). Note: this experiment was done on our best performing model different from the one used for annotations.

6.2 Reverse Models

Our choice of reverse model was directed by the best parameters from the Forward models. We compared the BLEU score for Greedy and Beam search in the reverse predictions. We thus chose a model having 4 layers in both encoder and decoder.

Model	Search	Parameter	BLEU
Enc4Dec4	Greedy	-	87.34
Enc4Dec4	B5/10/20	LP0	87.35
Enc1Dec2	Greedy	-	86.68
Enc1Dec2	B5/10/20	LP0	86.8

Table 6.6: Higher BLEU score (on Dev set) in the reverse direction can be attributed to the fact that the reverse direction is a comparatively easier task because of the structured nature of source MR. Beam width does not impact our results (thus, represented in the same cell). Here Enc1Dec2 represents 1 layer in Encoder and 2 layers in Decoder.

We did a similar human evaluation (by Annotator 1 only) on these re-ranked predictions to get a better understanding of the model and compared the statistics. Of the earlier 50 oracles found at the first place, our re-ranking procedure still maintain them as the oracles without changing their order in the n-best list. We had anticipated that for 18 cases which earlier had potential oracles in the n-best list, our approach would be able to move these oracle predictions at the first place. However our model re-ranked 6 oracles out of these 18 cases. Thus, there still remained 12 cases where our model could not move further up these predictions at the top of the list. This we found was due to the fact that while forward model tends to omit information, our reverse model surprisingly tends to add the very same information in the reconstructions. On the other hand, for a perfect solution, the reconstruction was also perfect, where none of the oracles were moved lower in the list. This phenomena seems unintuitive to us at this stage. To our surprise, the results also degraded in terms of BLEU. Thus, we considered moving towards using a classifier based approach to detect errors in terms of *Semantic Adequacy*.

Oracle [1/0/-1]	After Forward/Before Re-ranking	After Re-ranking
1	50	56
0	18	12
-1	32	32

Table 6.7: We used the same notation and same set of 100 examples as in Table 6.2 to compare our reverse model. The previously detected 50 oracles at the first position (denoted by 1) were not shifted down in their position and still captured by the re-ranking procedure. 6 of the earlier probable oracles were now re-ranked at the first position. Thus, our model slightly improved results in terms of oracle detection.

Model	BLEU
Forward Model	24.6
Re-ranking Model	24.52

Table 6.8: Re-ranked predictions (with the reverse model) have a slightly lesser BLEU score, a setback to our approach of using this model and edit distance as the score.

— 7 —
Conclusion and Future Work

7.1 Conclusion

We employed the open source *tf-seq2seq* framework for training a char2char model on the *E2E NLG Challenge* data. This could be done with limited effort, without requiring delexicalization, lowercasing or even tokenization, by exploiting standard options provided by the framework as well as the simplicity of the model.

Human annotators found the predictions to have great linguistic quality, somewhat to our surprise, but also confirming the observations in Karpathy (2015). On the adequacy side, omissions were the major drawback; no hallucinations were observed and only very few instances of repetition. Even though temperature sampling performs better and produce more diverse predictions as opposed to greedy and even traditional beam search (with length penalty 0), using length penalty equal to 1, produced better utterances, in terms of BLEU. The presence of an *oracle* in the top-20 predictions (more than 70 % of the cases), which we found using Human Evaluation, motivated us towards further employing re-ranking techniques.

7.2 Future Work

The first and foremost step would be to re-run some of our best models for the recently updated (according to our recommendations) E2E NLG Challenge dataset. During the next couple of weeks, we are going to explore two different mechanisms to handle omissions in the generated utterances.

7.2.1 Re-ranking

For the next steps, we would like to revisit our reverse model for re-ranking with a more in-depth analysis of the produced reverse constructions. This could also help us understand the data better, identifying the possible bias which leads to omission in the forward predictions at the first place.

Parallelly, we will train the Siamese based classifier as described in 4 which in itself is a novel mechanism to detect omissions from the similarity between the hidden representations of the MR and the predicted utterance.

7.2.2 Reinforcement Learning

Natural Language Generation can also be considered in a *Reinforcement Learning* paradigm. Re-ranking techniques, that we proposed earlier, rely upon the presence of an *oracle* in the top-k probable candidates for each MR. This creates a bottleneck when this *oracle* prediction is further down the n-best list. A natural extension to the seq2seq model is using Reinforcement Learning approaches which could guide the model itself during the training phase. The recent sequence tutor model (Jaques et al., 2016) can be a promising direction in this regard.

— A —
Appendix

Slots	Selected Samples per slot	
3	MR	name[Blue Spice], priceRange[£20-25], area[riverside]
	RF	*Blue Spice has items in the £20-25 price range and is in riverside.*
	Pred	Blue Spice is located in the riverside area with a price range of £20-25.
4	MR	name[The Punter], customer rating[high], area[riverside], kidsFriendly[yes]
	RF	*In riverside area, there is The Punter, which is high rated by customers and kids are friendly.*
	Pred	The Punter is a kid friendly restaurant in the riverside area with a high customer rating.
5	MR	name[Green Man], eatType[pub], food[English], area[city centre], near[Cafe Rouge]
	RF	*Green Man is a pub that can be found in the city centre, near caffe Rouge and serves English-style food.*
	Pred	Green Man is an English pub located in the city centre near caffe Rouge.
6	MR	name[The Golden Palace], eatType[coffeee shop], food[Japanese], priceRange[£20-25], **customer rating[high]**, area[riverside]
	RF	*For highly-rated Japanese food pop along to The Golden Palace coffee shop. Its located on the riverside. Expect to pay between 20-25 pounds per person.*
	Pred	The Golden Palace is a coffee shop providing Japanese food in the £20-25 price range. It is located in the riverside area.
7	MR	name[The Rice Boat], food[Chinese], priceRange[cheap], customer rating[average], area[city centre], **familyFriendly[no]**, near[Express by Holiday Inn]
	RF	*The Rice Boat is a not family friendly,cheap, average rated Chinese food restaurant near Express by Holiday Inn.*
	Pred	The Rice Boat provides Chinese food in the cheap price range. It is located in the city centre near Express by Holiday Inn. Its customer rating is average.
8	MR	name[The Eagle], eatType[coffeee shop], food[Japanese], priceRange[moderate], customer rating[1 out of 5], area[riverside], kidsFriendly[yes], near[Burger King]
	RF	*There is a one star mid priced family friendly coffee shop The Eagle near Burger King in the City centre. It offers Chinese food.*
	Pred	The Eagle is a kid friendly Japanese coffee shop in the riverside area near Burger King. It has a moderate price range and a customer rating of 1 out of 5.

Table A.1: Example utterances produced by the model for the selected samples of different number of slots. MR represents Meaning Representation based dialogue act. RF (in italics) represents the references and Pred is the the most probable realization from a beam of 20. Omissions of semantic material are highlighted in bold. We show the first example for each slot arity in a systematic manner in the order they appear.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Britz, D. (2015). Wildml: Ai, deep learning and nlp. `http://www.wildml.com/`.

Britz, D., Goldie, A., Luong, T., and Le, Q. (2017). Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

Chisholm, A., Radford, W., and Hachey, B. (2017). Learning to generate one-sentence biographies from wikidata. *arXiv preprint arXiv:1702.06235*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Dušek, O. and Jurčíček, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

Goyal, R., Dymetman, M., and Gaussier, E. (2016). Natural Language Generation through Character-based RNNs with Finite-State Prior Knowledge. In *Proc. COLING*, Osaka, Japan.

Graves, A. (2012). Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE.

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

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. (2016). Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. *See https://arxiv. org/abs/1611.02796 v8*.

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lampouras, G. and Vlachos, A. (2016). Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112. The COLING 2016 Organizing Committee.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015a). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015b). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Mei, H., Bansal, M., and Walter, M. R. (2015). What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Mueller, J. and Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Neculoiu, P., Versteegh, M., Rotaru, M., and Amsterdam, T. B. (2016). Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148.

Neubig, G. (2017). Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

Novikova, J., Dušek, O., and Rieser, V. (2017). The E2E NLG Shared Task.

Novikova, J., Lemon, O., and Rieser, V. (2016). Crowd-sourcing NLG Data: Pictures Elicit Better Data. *arXiv preprint arXiv:1608.00339*.

Olah, C. (2015). Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs`.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tillmann, C. and Ney, H. (2003). Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics*, 29(1):97–133.

Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015). Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542.

Vinyals, O. and Le, Q. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xiao, C., Dymetman, M., and Gardent, C. (2016). Sequence-based structured prediction for semantic parsing. *Proceedings Association For Computational Linguistics, Berlin*.

Zhao, S. and Zhang, Z. (2016). An efficient character-level neural machine translation. *arXiv preprint arXiv:1608.04738*.

A surprisingly effective out-of-the-box char2char model on the *E2E NLG Challenge* dataset

Anonymous submission

Abstract

We train a char2char model on the *E2E NLG Challenge* data, by exploiting "out-of-the-box" the recently released *tf-seq2seq* framework, using some of the standard options offered by this tool. With minimal effort, and in particular without delexicalization, tokenization or lowercasing, the obtained raw predictions, according to a small scale human evaluation, are excellent on the linguistic side and quite reasonable on the adequacy side, the primary downside being the possible omissions of semantic material. However, in a significant number of cases (more than 70%), a perfect solution can be found in the top-20 predictions, indicating promising directions for solving the remaining issues.

1 Introduction

Very recently, researchers (Novikova et al., 2017) at Heriot-Watt University proposed the E2E NLG Challenge[1] and released a dataset consisting of 50K (MR, RF) pairs, MR being a slot-value Meaning Representation of a restaurant, RF (human ReFerence) being a natural language utterance rendering of that representation. The utterances were crowd-sourced based on pictorial representations of the MRs, with the intention of producing more natural and diverse utterances compared to the ones directly based on the original MRs (Novikova et al., 2016).

Most of the RNN-based approaches to Natural Language Generation (NLG) that we are aware of, starting with (Wen et al., 2015), generate the output word-by-word, and resort to special delexicalization or copy mechanisms (Gu et al., 2016) to handle rare or unknown words, for instance restaurant names or telephone numbers. One exception is (Goyal et al., 2016), who employed a char-based seq2seq model where the input MR is simply represented as a character sequence, and the output is also generated char-by-char; this approach avoids the rare word problem, as the character vocabulary is very small.

While (Goyal et al., 2016) used an additional finite-state mechanism to guide the production of well-formed (and input-motivated) character sequences, the performance of their basic char2char model was already quite good. We further explore how a recent out-of-the box seq2seq model would perform on the E2E NLG Challenge, when used in a char-based mode. We choose the attention-based *tf-seq2seq* framework provided by the authors of (Britz et al., 2017) (which we detail in the next section).

Using some standard options provided by this framework, and without any pre- or post-processing (not even tokenization or lowercasing), we obtained results on which we conducted a small-scale human evaluation on one hundred MRs, involving two evaluators. This evaluation, on the one hand, concentrated on the linguistic quality, and on the other hand, on the semantic adequacy of the produced utterances. On the linguistic side, vast majority of the predictions were surprisingly grammatically perfect, while still being rather diverse and natural. In particular, and contrary to the findings of (Goyal et al., 2016) (on a different dataset), our char-based model never produced non-words. On the adequacy side, we found that the only serious problem was the tendency (in about half of the evaluated cases) of the model to omit to render one (rarely two) slot(s); on the other end, it never hallucinated, and very rarely duplicated, material. To try and assess the

[1] `http://www.macs.hw.ac.uk/InteractionLab/E2E/`

potential value of a simple re-ranking technique (which we did not implement at this stage, but the approach of (Wen et al., 2015) and more recently the "inverted generation" technique of (Chisholm et al., 2017) could be used), we generated (using the beam-search option of the framework) 20-best utterances for each MR, which the evaluators scanned towards finding an "oracle", i.e. a generated utterance considered as perfect not only from the grammatical but also from the adequacy viewpoint. An oracle was found in the first position in around 50% of the case, otherwise among the 20 positions in around 20% of the cases, and not at all inside this list in the remaining 30% cases. On the basis of these experiments and evaluations we believe that there remains only a modest gap towards a very reasonable NLG seq2seq model for the E2E NLG dataset.

2 Model

Our model is a direct use of the seq2seq open-source software framework[2], built over TensorFlow (Abadi et al., 2016), and provided along with (Britz et al., 2017), with some standard configuration options that will be detailed in section 3. While in their large-scale NMT experiments (Britz et al., 2017) use word-based sequences, in our case we use character-based ones. This simply involves changing the "delimiter" option in the configuration files.

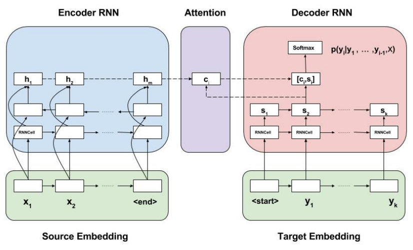


Figure 1: The seq2seq architecture of (Britz et al., 2017) (drawing borrowed from that paper). Contrary to word-based sequences, we use character-based sequences for generating grammatically correct and natural utterances.

Figure 1, borrowed from (Britz et al., 2017), provides an overview of the framework. While many options are configurable (number of layers, unidirectional *vs* bidirectional encoder, additive *vs* multiplicative attention mechanism, GRU (Cho et al., 2014) *vs* LSTM cells (Hochreiter and Schmidhuber, 1997), etc.), the core architecture is common to all models. This is by now a pretty standard attention-based encoder-decoder architecture based on (Bahdanau et al., 2015; Luong et al., 2015). The encoder RNN embeds each of the source words (in our case, characters) into vectors exploiting the hidden states computed by the RNN. The decoder RNN predicts the next word (resp. character) based on its current hidden state, previous character, and also based on the "context" vector c_i, which is an attention-based weighted average of the embeddings of the source words (resp. characters).

[2] https://github.com/google/seq2seq.

3 Experiments

3.1 Dataset

(Novikova et al., 2016) explain the protocol followed for crowdsourcing the *E2E NLG Challenge* dataset. Slightly different from the description in the article, there are two additional slots in the dataset: 'kidsFriendly' and 'children-friendly' which seem to be alternates for 'familyFriendly'. Thus, there are in total 10 slots (in decreasing order of frequency of being mentioned in the dataset MRs): name (100%), food (83%), customer rating (68%), priceRange (68%), area (60%), eatType (51%), near (50%), familyFriendly (25%), kidsFriendly (19%), children-friendly (19%). Also, the number of active slots in the MRs varies as: 3 (5%), 4 (17%), 5 (19%), 6 (19%), 7 (16%), 8 (4%).

3.2 Implementation

The *tf-seq2seq* toolkit (Britz et al., 2017) trains on pairs of sequences presented in parallel text format (separate source and target sequence files).[3]

Taking cue from the recommended configurations in Table 7 of (Britz et al., 2017) and the provided example configs in *tf-seq2seq*, we mainly experimented with different numbers of layers in the encoder and decoder as well as different beam widths, while using the bi-directional en-

[3] We cleaned the E2E NLG Challenge data as there were a few erroneous newline characters (Line 603 in devset.csv as well as 30048 in trainset.csv). There were different character encodings for MR and RF, which we uniformized to utf-8. Also, there were a few wrongly encoded characters (such as on line 23191 in trainset.csv). We normalized these characters, after which there remained only two non-ascii characters: £ and é. With the final version of the paper we will release the code for processing of the data, conversion to parallel text format as well as our configuration files for the tf-seq2seq model.

Model Specification	Beam Width	Length Penalty	Depth				
Encoder			1	1	2	4	4
Decoder			1	2	2	4	4
Cell Unit			GRU	GRU	GRU	GRU	LSTM
Beam Search	No Beam		20.94	22.59	23.5	23.84	23.98
	Beam 5	LP0	15.85	22.47	21.76	22.73	20.15
	Beam 10	LP0	14.5	21.4	19.98	21.15	18.88
	Beam 20	LP0	13.48	20.18	18.5	19.94	17.93
	Beam 5	LP1	20.64	24.77	24.67	**24.94**	23.87
	Beam 10	LP1	21	25.05	24.88	24.69	24.27
	Beam 20	LP1	21.27	25.4	24.96	**24.6**	24.05

Table 1: BLEU scores on devset while varying the depth of both encoder and decoder RNNs, type of cell unit, different beam widths and length penalty. (Results reported for a single experiment with training and prediction.)

coder along with the "additive" attention mechanism. As also observed by Britz et al. (2017), using a non-null "length-penalty" (alias length normalization (Wu et al., 2016)), significantly improved the decoding results.

3.3 Results

We report the BLEU scores[4] for different configurations of the seq2seq model in Table 1. In our initial experiments, using a beam-width of 5 (with no length penalty), with 4 layers in both the encoder and decoder and GRU cells, showed the best results in terms of BLEU (score of 24.94).

We observed significant improvements using a length penalty of 1, and decided to use this architecture as a basis for the human evaluations, with a beam-width of 20 to facilitate the observation of oracles. These evaluations were thus conducted on [encoder 4 layers, decoder 4 layers, GRU cell, beam-width 20, length penalty 1], though we found slightly better performing models in terms of BLEU at a later stage.

4 Evaluation

The human evaluations were performed by two annotators on the top 20 predictions of the previously discussed model, for the first 100 MRs of the devset, using the following metrics:

1. Semantic Adequacy
   a) Omission [1/0]: information present in the MR that is omitted in the predicted utterance (1=No omission, 0=Omission). **b) Addition [1/0]:** information in the predicted utterance that is absent in the MR (1=No addition, 0=Addition). **c) Repetition [1/0]:** repeated information in the predicted utterance (1=No repetition, 0=Repetition).

2. Linguistic Quality
   a) Grammar [1/0]: (1=Grammatically correct, 0=incorrect). Note: one annotator punished the model even for (rare) mistakes of punctuation. **b) Naturalness [2/1/0]:** subjective score to measure the naturalness of the utterance (2 being best). **c) Comparison to reference [1/0/-1]:** subjective score comparing the prediction with the crowdsourced RF. ('vsRef' in the Table 2, 1=Prediction better than RF, 0=Prediction at par with RF, -1=RF better than prediction).

3. Oracle [1/0/-1]: 1 if the first prediction is an "oracle" (i.e. considered as perfect, see section 1), 0 when the oracle is found in the top 20, and -1 when no oracle is found there.

5 Analysis

We show a few examples of utterances (predictions in first position, i.e. most probable) produced by our model, for discussion.[5]

1. **[MR]:** name[The Punter], customer rating[high], area[riverside], kidsFriendly[yes]
   [RF]: *In riverside area, there is The Punter, which is high rated by customers and kids are friendly.*
   [Pred]: The Punter is a kid friendly restaurant in the riverside area with a high customer rating.

2. **[MR]:** name[The Golden Palace], eatType[coffeee shop], food[Japanese], priceRange[£20-25], customer rating[high], area[riverside]
   [RF]: *For highly-rated Japanese food pop along to The Golden Palace coffee shop. Its located on the riverside. Expect to pay between 20-25 pounds per person.*
   [Pred]: The Golden Palace is a coffee shop providing Japanese food in the £20-25 price range. It is located in the riverside area.

[4]Calculated using the multi-bleu perl script bundled with *tf-seq2seq*.

[5]We release the full list of human annotated examples, including the 20-best predictions and oracles, in the supplementary material accompanying this submission.

Ann	O(1/0)	A(1/0)	R(1/0)	G(1/0)	N(2/1/0)	vsRef(1/0/-1)	Or(1/0/-1)
Ann 1	51/49	100/0	97/3	93/7	85/13/2	46/16/38	50/18/32
Ann 2	51/49	100/0	98/2	98/2	80/18/2	29/36/35	51/18/31
Mean	51/49	100/0	97.5/2.5	95.5/4.5	82.5/15.5/2	37.5/26/36.5	50.5/18/31.5

Table 2: Human annotations for 100 samples using different metrics defined in Sec. 4. O (Omission), A (Addition), R (Repetition) and G (Grammar) are on binary scale. Naturalness is measured as (2/1/0) and Oracle as (1/0/-1). Predictions were also judged against the reference on a scale of (1/0/-1).

Slots	DA	Or@1	Or	No Or
3	1(1%)	1(100%)	0(0%)	0(0%)
4	29(29%)	24(83%)	3(10%)	2(7%)
5	25(25%)	13(48%)	6(24%)	6(28%)
6	29(29%)	11(34%)	5(17%)	13(48%)
7	11(11%)	1(9%)	3(27%)	7(64%)
8	5(5%)	1(20%)	1(20%)	3(60%)
Total	100	51	18	31

Table 3: Human annotations for different slots using beam-width 20. 'Or@1' represents the presence of an 'oracle' in first position while 'Or' represents the presence of 'Oracle' (desirable) in the top-20 predictions. Cases where no oracle was found are marked as 'No Or'.

3. **[MR]:** name[Strada], food[Fast food], priceRange [moderate], customer rating[1 out of 5], kidsFriendly[no], near [Rainbow Vegetarian Cafe]
   [RF]: *Strada is a Fast food restaurant near the Rainbow Vegetarian caffe which has a moderate customer rating of 1 out of 5 for a non Kids friendly restaurant*
   [Pred]: Strada is a moderately priced fast food restaurant in the **moderate price range**. It is located near Rainbow Vegetarian caffe.

Among the utterances produced by the model in first position (Pred), the most prominent issue was that of omissions (underlined in example 2). There were no additions or non-words (which was one of the primary concerns for (Goyal et al., 2016)). We observed only a couple of repetitions which were actually accompanied by omission of some slot(s) in the same utterance (repetition highlighted in bold in example 3). Surprisingly enough, we observed a similar issue of omissions in human references (target for our model). We then decided to perform comparisons against the human reference ('vsRef' in Table 2). Often, the predictions were found to be semantically or grammatically better than the human reference; for example observe the underlined portion of the reference in the first example. The two annotators independently found the predictions to be mostly grammatically correct as well as natural (to a slighty lesser extent).[6]

[6]Annotator-1 was more severe in highlighting even the (rare) punctuation issues as grammatical mistakes. There was also a slight disagreement with Annotator-2 being more severe than Annotator-1 when assessing the references against the predictions.

A general feeling of the annotators was that the predictions, while showing a significant amount of linguistic diversity and naturalness, had a tendency to respect grammatical constraints *better* than the references; the crowdsourcers tended to strive for creativity, sometimes not supported by evidence in the MR, and often with little concern for linguistic quality; it may be conjectured that the seq2seq model, by "averaging" over many linguistically diverse and sometimes incorrect training examples, was still able to learn what amounts to a reasonable linguistic model for its predictions.

We also investigate whether we could find an 'oracle' (perfect solution as defined in section 1) in the top-20 predictions and observed that in around 70% of our examples the oracle could be found in the top results (see Table 3), very often (51%) at the first position. In the rest 30% of the cases, even the top-20 predictions did not contain an oracle. We found that the presence of an oracle was dependent on the number of slots in the MR. When the number of slots was 7 or 8, the presence of an oracle in the top predictions decreased significantly to approximately 40%. In contrast, with 4 slots, our model predicted an oracle right at the first place for 83% of the cases.

6 Conclusion

We employed the open source *tf-seq2seq* framework for training a char2char model on the *E2E NLG Challenge* data. This could be done with minimal effort, without requiring delexicalization, lowercasing or even tokenization, by exploiting standard options provided with the framework.

Human annotators found the predictions to have great linguistic quality, somewhat to our surprise, but also confirming the observations in (Karpathy, 2015). On the adequacy side, omissions were the major drawback; no hallucinations were observed and only very few instances of repetition. We hope our results and annotations can help understand the dataset and issues better, while also being useful for researchers working on the challenge.

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 Mane, 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 Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR abs/1603.04467* .

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

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *CoRR abs/1703.03906* .

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR abs/1702.06235* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. EMNLP*.

Raghav Goyal, Marc Dymetman, and Eric Gaussier. 2016. Natural Language Generation through Character-based RNNs with Finite-State Prior Knowledge. In *Proc. COLING*. Osaka, Japan.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. ACL*.

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

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2016. Character-based neural machine translation. In *Proc. ICLR*. pages 1–11.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E NLG Shared Task .

Jekaterina Novikova, Oliver Lemon, and Verena Rieser. 2016. Crowd-sourcing NLG Data: Pictures Elicit Better Data. *CoRR abs/1608.00339* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*. pages 3104–3112.

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

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klinger, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR abs/1609.08144* .

Appendix A. Sample predictions

Slots	Selected Samples per slot	
3	MR	name[Blue Spice], priceRange[£20-25], area[riverside]
	RF	*Blue Spice has items in the £20-25 price range and is in riverside.*
	Pred	Blue Spice is located in the riverside area with a price range of £20-25.
4	MR	name[The Punter], customer rating[high], area[riverside], kidsFriendly[yes]
	RF	*In riverside area, there is The Punter, which is high rated by customers and kids are friendly.*
	Pred	The Punter is a kid friendly restaurant in the riverside area with a high customer rating.
5	MR	name[Green Man], eatType[pub], food[English], area[city centre], near[Cafe Rouge]
	RF	*Green Man is a pub that can be found in the city centre, near caffe Rouge and serves English-style food.*
	Pred	Green Man is an English pub located in the city centre near caffe Rouge.
6	MR	name[The Golden Palace], eatType[coffeee shop], food[Japanese], priceRange[£20-25], **customer rating[high]**, area[riverside]
	RF	*For highly-rated Japanese food pop along to The Golden Palace coffee shop. Its located on the riverside. Expect to pay between 20-25 pounds per person.*
	Pred	The Golden Palace is a coffee shop providing Japanese food in the £20-25 price range. It is located in the riverside area.
7	MR	name[The Rice Boat], food[Chinese], priceRange[cheap], customer rating[average], area[city centre], **familyFriendly[no]**, near[Express by Holiday Inn]
	RF	*The Rice Boat is a not family friendly,cheap, average rated Chinese food restaurant near Express by Holiday Inn.*
	Pred	The Rice Boat provides Chinese food in the cheap price range. It is located in the city centre near Express by Holiday Inn. Its customer rating is average.
8	MR	name[The Eagle], eatType[coffeee shop], food[Japanese], priceRange[moderate], customer rating[1 out of 5], area[riverside], kidsFriendly[yes], near[Burger King]
	RF	*There is a one star mid priced family friendly coffee shop The Eagle near Burger King in the City centre. It offers Chinese food.*
	Pred	The Eagle is a kid friendly Japanese coffee shop in the riverside area near Burger King. It has a moderate price range and a customer rating of 1 out of 5.

Table 1: Example utterances produced by the model for the selected samples of different number of slots. MR represents Meaning Representation based dialogue act. RF (in italics) represents the references and Pred is the the most probable realization from a beam of 20. Omissions of semantic material are highlighted in bold. We show the first example for each slot arity in a systematic manner in the order they appear.