End-to-End Content and Plan Selection for Natural Language Generation

E2E NLG Challenge - HarvardNLP system

Sebastian Gehrmann
Harvard SEAS
gehrmann@seas.harvard.edu

Falcon Z. Dai
TTI-Chicago
dai@ttic.edu

Henry Elder
ADAPT
henry.elder@adaptcentre.ie

Alexander M. Rush
Harvard SEAS
srush@seas.harvard.edu

Abstract

This paper describes our entry for the INLG 2018 E2E NLG challenge. Generating fluent natural language descriptions from structured data is a key sub-task for conversational agents. In the E2E NLG challenge, the task is to generate these utterances conditioned on multiple attributes and values. Our system utilizes several extensions to the general-purpose sequence-to-sequence (S2S) architecture to model the latent content selection process, particularly different variants of copy attention and coverage decoding. In addition, we propose a new training method based on diverse ensembling to encourage the model to learn latent plans in training. We empirically evaluate these techniques and show that the system increases the quality of generated text across five automated metrics. Out of a total of sixty submitted systems from 16 institutions, our best system ranks first-place in three of the five metrics, including ROUGE.

1 Introduction

Recent developments in end-to-end learning with neural networks have significantly advanced the state-of-the-art in text transduction tasks, such as machine translation. These techniques have also enabled researchers to develop systems for generating textual output from complex non-textual inputs such as images and tables. End-to-end methods may enable the creation of general-purpose conditional text-generation models, and in recent years they have been applied to increasingly complex data to simultaneously learn sentence planning and surface realization (Wen et al., 2015; Mei et al., 2015; Dušek and Jurčíček, 2016; Lampouras and Vlachos, 2016).

The E2E NLG dataset (Novikova et al., 2017) provides a new test bed for exploring the frontier of automatic text generation. The dataset provides examples where an input, expressed as a dialogue act-based meaning representation (MR), is aligned to on average 8.1 references. An example is shown in Figure 1. The associated E2E challenge prompts teams to develop a model that learns the semantic alignment from MR to utterance, and to generate a utterances that are similar to reference texts, and highly rated by humans.

MR	name[The Golden Palace], eatType[coffee shop], food[Fast food], priceRange[cheap], customer rating[5 out of 5], area[riverside]
Reference	A coffee shop located on the riverside called The Golden Palace, has a 5 out of 5 customer rating. Its price range are fairly cheap for its excellent Fast food.

Figure 1: An example MR and utterance from the E2E NLG dataset. Each example consists of a set of key-value pairs and a natural language description.

The report describes our entry to the E2E Generation Challenge. We first survey how extensions to S2S models with attention mechanism (Sutskever et al., 2014; Bahdanau et al., 2014) perform in this task. These include: a mechanism to copy words from the input into the generated text (Vinyals et al., 2015), a coverage and a length penalty (Tu et al., 2016) to ensure that the model takes into account all of the input, and the impact of changing the model architecture itself from bidirectional LSTMs to the recently introduced Transformer model (Vaswani et al., 2017). We show that the extensions lead to an improvement to metrics and further allow the system to model the content selection aspect of the NLG problem.

Furthermore, we note that the training data contains different types of responses to similar inputs. These include detailed responses, but also some incoherent fragments (*Not family friendly Alimentum across from Burger King*). In order to develop a model that is robust to different response styles, we employ a diverse training technique (Guzman-Rivera et al., 2012) to learn multiple submodels, implicitly modeling different types of responses. This method effectively partitions out the training data during the process of training the model itself, i.e. end-to-end. We show that the technique can lead to a model that picks a specific type of response to a given query.

Our final system increases the quality of generated text across five different automated metrics (BLEU, NIST, METEOR, ROUGE, and CIDEr) over the TGEN baseline (Dušek and Jurčíček, 2016), a strong S2S model (Bahdanau et al., 2014) with a penalty for straying away from the input representation. Among the 60 submissions to the E2E NLG challenge, our system ranks first in METEOR, ROUGE, and CIDEr scores, third in BLEU, and sixth in NIST. Results of an evaluation of quality and naturalness using the TrueSkill algorithm (Sakaguchi et al., 2014) place our submission tied for second place with several other approaches.

2 Data for E2E NLG

Traditional approaches to natural language generation separate the generation of a sentence plan from the surface realization. First, an input is mapped into a format that represents the layout of the output sentence, for example an adequate pre-defined template. Then, the surface realization transforms the intermediary structure into text (Stent et al., 2004). First steps away from hand-crafted templates and rules aimed to learn with little dependence on heuristics, for example by using phrase-based language models during the generation (Oh and Rudnicky, 2000; Mairesse and Young, 2014). More recent work combines both steps by learning plan and realization jointly using end-to-end trained models (e.g. Wen et al., 2015). However, corpus-based approached are restricted by the limited availability of data.

In this work we use the crowd-sourced E2E NLG dataset by Novikova et al. (2017) providing 50,000 examples of MR and reference pairs in the restaurant domain. Each datapoint comprises a meaning representation of on average 5.43 attribute-value pairs, and a corresponding natural language utterance. A list of attributes and corresponding examples is shown in Table 1. The dataset is split into 76% training, and 9% validation , and 15% test data. The validation data is multi-reference with on average 8.1 references for each MR. An additional test set has 630 MR's and unseen utterances.

Attribute	Example
area	city centre, riverside, ...
customerRating	1 out of 5, average, ...
eatType	coffee shop, restaurant, ...
familyFriendly	yes / no
food	Chinese, English, ...
name	Wildwood, The Wrestlers, ...
near	Café Sicilia, Clare Hall, ...
priceRange	less than £20, cheap, ...

Table 1: List of possible attributes in the E2E NLG dataset and examples.

3 Sequence-to-Sequence Generation

Our main approach begins with a sequence-to-sequence (S2S) model for text transduction. In the standard text-to-text problems let $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}), \ldots (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}) \in (\mathcal{X}, \mathcal{Y})$ be a set of N aligned source and target sequence pairs, with $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ denoting the ith element in $(\mathcal{X}, \mathcal{Y})$ pairs. Further, let $\mathbf{x} = x_1, \ldots, x_m$ be the sequence of m tokens in the source, and $\mathbf{y} = y_1, \ldots, y_n$ the target sequence of length n. Let $\mathcal{V}$ be the vocabulary of possible tokens, and let $[n]$ denote the list of integers up to n, $[1, \ldots, n]$.

S2S aims to learn a function f parametrized by θ that maximizes the conditional probability of $p_\theta(\mathbf{y}|\mathbf{x})$. We further assume that the target is generated from left to right, such that $p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{n} p_\theta(y_t|\mathbf{y}_{[t-1]}, \mathbf{x})$. f takes the form of an encoder-decoder architecture with attention, in which an encoder-function e encodes the source and a decoder function d generates the target from the output of e, such that $f(\mathbf{x}) = d(e(\mathbf{x}))$. Training uses stochastic gradient descent to minimize the negative log-likelihood of the training data.

In order to apply E2E models, we need to transform the given MR into a sequence of tokens. This is not trivial, since not all attributes appear at all times, and each attribute might have multi-token values, such as *area[city centre]*. In our approach, we introduce special start and stop tokens for *each* of the attributes, and use these to mark the boundaries of its value; for example, an attribute *area[city centre]* becomes *__start_area__ city centre __end_area__*. We then concatenate these fragments into a long sequence to represent the original MR as an input sequence to our models. In this approach, restaurant names were not delexicalized. We note that this approach works best if the attributes always appear in the same order.

3.1 Attention Mechanisms

We model both encoder and decoder as bidirectional LSTM (Hochreiter and Schmidhuber, 1997), where c_i represents the hidden state of the encoder LSTM at timestep i, and h_t represents the hidden state of the decoder at timestep t. Let $\mathrm{d_{hid}}$ denote the size of the hidden state of the decoder.

We investigate two different calculations of the attention distribution. The first formulation follows Bahdanau et al. (2014) and uses an multilayer perceptron. Given a timestep t and the softmax function sm, the attention distribution a^t is defined such that

$$e_i^t = v^T \tanh(W_c c_i + W_h h_t + b_{\text{attn}})$$
$$a^t = \mathrm{sm}(e^t),$$

where $v \in \mathbb{R}^{\mathrm{d_{hid}}}$, $W_c \in \mathbb{R}^{\mathrm{d_{hid}} \times \mathrm{d_{hid}}}$, $W_h \in \mathbb{R}^{\mathrm{d_{hid}} \times \mathrm{d_{hid}}}$, and $b_{\text{attn}} \in \mathbb{R}^{\mathrm{d_{hid}}}$ are all trainable parameters. Given the attention distribution, the context vector is a sum of hidden states in the encoder, weighted by its attention

$$c_t^* = \sum_i a_i^t c_i.$$

Let $[h_t, c_t^*]$ denote the concatenation of hidden state and context vector. This can be used to calculate an intermediate representation $p_{\text{out}} \in \mathbb{R}^{\mathrm{d_{hid}}}$ as

$$p_{\text{out}} = W_{\text{out}}[h_t, c_t^*] + b_{\text{out}}.$$

Finally, a generator calculates the conditional probability distribution over the vocabulary $\mathcal{V}$

$$p(y_t | \mathbf{y}_{[t-1]}, \mathbf{x}) = \mathrm{sm}(W_{\text{gen}} p_{\text{out}} + b_{\text{gen}}).$$

$W_{\text{out}} \in \mathbb{R}^{2 \cdot \mathrm{d_{hid}} \times \mathrm{d_{hid}}}$, $b_{\text{out}} \in \mathbb{R}^{\mathrm{d_{hid}}}$, $W_{\text{gen}} \in \mathbb{R}^{\mathrm{d_{hid}} \times |\mathcal{V}|}$, and $b_{\text{gen}} \in \mathbb{R}^{|\mathcal{V}|}$, are trainable parameters.

We additionally explore a second formulation for the attention mechanism, first introduced by Luong et al. (2015), that includes two changes to this calculation. The most notable change is that the calculation of the attention distribution no longer uses an MLP. Instead, the dot product between the hidden states of encoder and decoder $e_i^t = h_t c_i^T$ is applied. The other change to the previously described procedure is that p_{out} includes an additional nonlinearity

$$p_{\text{out}} = \tanh(W_{\text{out}}[h_t, c_t^*] + b_{\text{out}}).$$

Hereafter, we will refer to the two types of attention as *MLP* and *dot*.

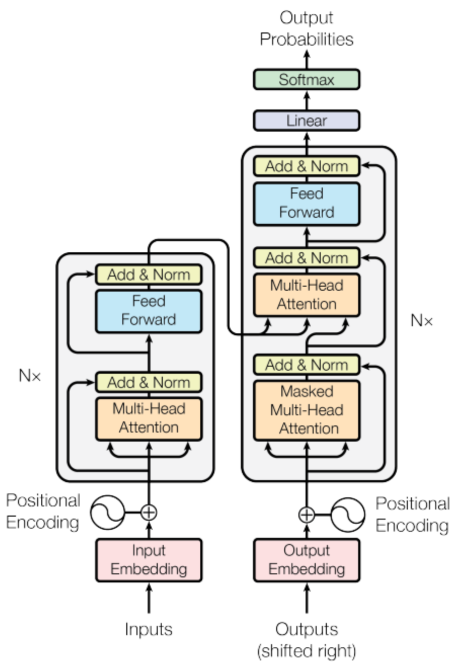

Figure 2: Replication of an illustration of the Transformer S2S model by Vaswani et al. (2017).

3.2 Transformer Model

We give a brief overview of the Transformer model and refer to the original paper by Vaswani et al. (2017) for an in-depth description. The premise of this model is that recurrent neural networks can be replaced as encoder and decoders. The Transformer only uses attention over the embeddings of inputs and outputs. An overview of the model is shown in Figure 2. The model is composed of multiple layers, each comprising two parts – a multi-head self-attention and a position-wise feed-forward network. Each part additionally has a residual connection (He et al., 2016) followed by a layer normalization (Ba et al., 2016). The last addition is a positional encoding of the input.

The first change from the attention described so far is that this model uses multiple heads, each of which uses a different linear projection of the input. Each attention uses the *dot* attention with the change that e_i^t is scaled by the root of the dimension of the vectors

$$e_i^t = \frac{h_t \cdot c_i^T}{\sqrt{\mathrm{d_{hid}}}}.$$

Then, all attentions are concatenated and linearly transformed into $\mathbb{R}^{\mathrm{d_{hid}}}$. Each layer of the model contains a feed-forward network that is applied to every single position individually, similar to a convolutional neural network with a filter-width of 1. The network uses two layers with a ReLu activation function

$$f(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

However, without any recurrence, there is no positional information in the model. To address this, the Transformer uses a positional encoding based on the sine and cosine functions

$$PE(pos, 2i) = \sin \frac{pos}{10000^{2i/\mathrm{d_{emb}}}}$$
$$PE(pos, 2i+1) = \cos \frac{pos}{10000^{2i/\mathrm{d_{emb}}}}$$

Here, pos is the position within a sequence, i is the dimension, and $\mathrm{d_{emb}}$ is the size of the word embedding. The resulting vector is added to the word embedding. The function is chosen since it represents a geometric progression from 2π to $20{,}000\pi$ which allows the model to learn to attend by relative positions. The new representations for each position in the text are then used as context for the attention layer.

4 Copy and Coverage

We additionally extend the S2S system based on ideas that have been shown to be useful for the related problem of sentence summary. In particular we implement the pointer-generator network similar to that introduced by Nallapati et al. (2016) and further extended by See et al. (2017).

Copy Model The copy model introduces a binary variable z_t for each decoding step t that acts as a switch between copying from the source and generating words. We model the joint probability following the procedure described by Gulcehre et al. (2016) as

$$p(y_t, z_t | y_{[t-1]}, \mathbf{x}) = \sum_{z \in \{0,1\}} p(y_t, z_t = z | \mathbf{y}_{[t-1]}, \mathbf{x})$$

To calculate the switching probability $p(z_t | \mathbf{y}_{[t-1]}, \mathbf{x})$, let $v \in \mathbb{R}^{\mathrm{d_{hid}}}$ be a trainable parameter. The hidden state of the decoder h_t is used to compute $p(z_t) = \sigma(h_t^T v)$ and decompose the joint distribution into two parts:

$$p(y_t | y_{[t-1]}, \mathbf{x}) = p(z_t = 1) p(y_t | z_t = 1) + p(z_t = 0) p(y_t | z_t = 0),$$

where every term is conditioned on $\mathbf{x}$ and $\mathbf{y}_{[t-1]}$. Here, $p(y_t | z_t = 0)$ is the distribution generated by the previously described S2S model, and $p(y_t | z_t = 0)$ is a distribution over $\mathbf{x}$ that is computed using the same attention mechanism (with separate parameters). In the E2E NLG case, all the values in the MR's should occur in the generated text and they are generally words that would not be generated by a language model. This allows us to follow Gulcehre et al. (2016) by assuming that every word that occurs in both source and target was copied to avoid having to marginalize over z. Then, we minimize the negative joint log-likelihood of y_t and z_t. This approach has the further advantage that it can handle previously unseen input by learning to copy these words into the correct position.

Coverage and Length Penalty We observed that generated text using vanilla S2S models with or without copy mechanism commonly omits some of the inputs. To prevent this, we use two penalty terms during inference; a length and a coverage penalty. We are using a coverage penalty during inference only, compared to Tu et al. (2016) who first introduced a coverage penalty term into the attention of an S2S model for neural machine translation and See et al. (2017) who used the same idea for abstractive summarization. Instead, we follow Wu et al. (2016) and introduce a penalty term cp defined as

$$cp(\mathbf{x}, \mathbf{y}) = \beta \cdot \sum_{i=1}^{|\mathbf{x}|} \log(\min(\sum_{t=1}^{|\mathbf{y}|} a_i^t, 1.0)).$$

Here, β is a parameter to control the strength of the penalty. This penalty term increases when too many generated words attend to the same input. We typically do not want to repeat the name of the restaurant or the type of food it serves. Thus, we only want to attend to the restaurant name once when we actually generate it. We also use the length penalty lp by Wu et al. (2016), defined as

$$lp(\mathbf{y}) = \frac{(5 + |\mathbf{y}|)^\alpha}{(5+1)^\alpha},$$

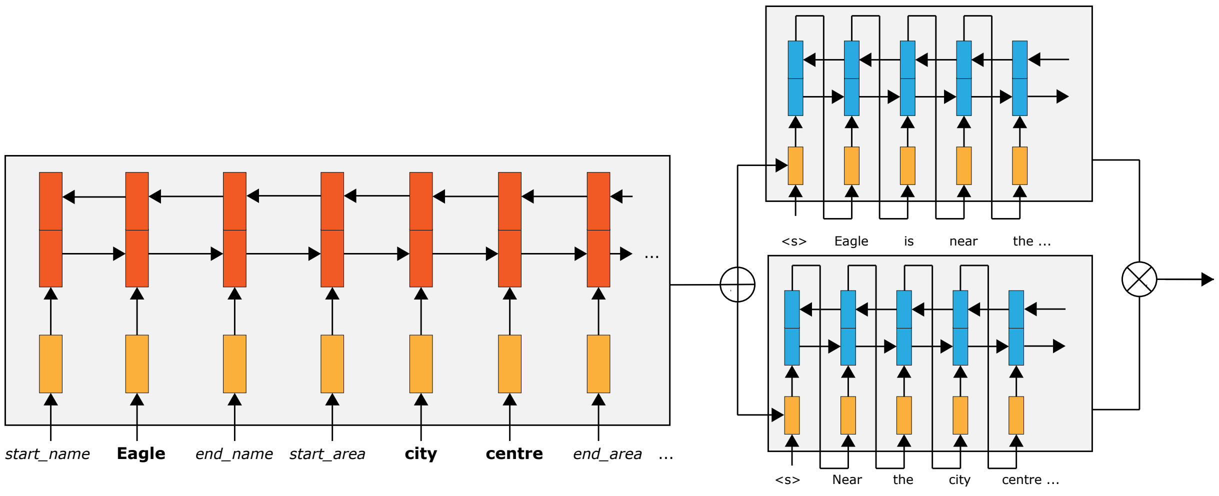

Figure 3: An illustration of the diverse ensembling method with $K = 2$ and a shared encoder. The encoder, shown on the left, reads the meaning representation and generates the context vectors. The context is then used in parallel by into the two separate decoders. Here, $\oplus$ represents the duplication of the input representation. The two decoders generate text independently from each other. Finally, only the decoder with the better generated text is chosen to receive a parameter update. The exclusive choice is illustrated by the $\otimes$ operation.

where α is a tunable parameter. The penalties are used to re-rank beams during the inference procedure such that the full score function s becomes

$$s(\mathbf{x}, \mathbf{y}, z) = \frac{\log p(\mathbf{y}, z|\mathbf{x})}{lp(\mathbf{y})} + cp(\mathbf{x}, \mathbf{y}).$$

5 Learning Latent Plans

Finally, we target an issue with this style of human based training data. Each sentence in the corpus follows an implicit latent sentence plan, that is independent of the input attributes. We observe that ignoring this variable leads to unnatural generated text, as if multiple underlying templates were combined into one. To more directly model the sentence planning step, we learn a mixture of models that operate over full sequences. Instead of training each part of the mixture on all the training data, we hope to implicitly cut out sentences that would divert from a possible plan.

The challenge here is to separate the training data, while simultaneously training different models that focus on one (or several) template structures. To find this segmentation, we assume that we have K models $f_1, \ldots, f_K$. These models can be completely disjoint or share a subset of their parameters (e.g. the word embeddings, the encoder, or both encoder and decoder). Following Guzman-Rivera et al. (2012), we introduce a random variable $w \sim \text{Cat}(1/K)$ that assigns a weight to each model for a given data point. The optimization target for each point becomes

$$\text{argmin}_{w,\theta} \sum_{i=1}^{|\mathcal{X}|} \sum_{k=1}^{K} w_k^i \cdot \ell(\mathbf{y}^i, f_k(\mathbf{x}^i)),$$

where $\ell(\mathbf{y}, \hat{\mathbf{y}})$ denotes the negative log-likelihood for one specific prediction. Thus, the optimization problem over the whole dataset is a joint optimization of assignments of models to data points and parameters to models.

In order to minimize the target, Guzman-Rivera et al. (2012) introduced the idea of a multiple-choice loss (MCL) to segment training data similar to a hard EM algorithm or k-Means clustering. With MCL, all K models are trained for one epoch. Then, each training point is assigned to the model with the minimal loss. After this segmentation, each model is trained for a further epoch using only the assigned data points. This process is repeated until the point assignments converge. Further work by Lee et al. (2016) reduce the computational overhead by introducing a stochastic MCL (sMCL) variant that does not require retraining. They define the E-Step as $\hat{k} = \text{argmax}_{k \in [K]} p_\theta(\mathbf{y}|\mathbf{x}, w = k)$. Setting $w_{\hat{h}}$ to 1 and all other entries in w to 0 achieves the segmentation. Following the objective above, only the model with the minimal negative log-likelihood is updated in the M-Step such that the problem becomes $\text{argmax}_\theta\, p_\theta(\mathbf{y}|\mathbf{x}, w)$.

We found that separating the models on the per-word loss leads to behavior where one of the mod-

Measure	Description
BLEU	Measures precision as ratio of correctly generated n-grams
NIST	Modification of BLEU, weights the n-grams rarity
METEOR	Aligns hypothesis to references and computes F-measure
ROUGE-L	Recall-based measure that finds longest common subsequence
CIDEr	Consensus based, compares similarity of n-grams to majority of references

Table 2: Short descriptions of automated metrics.

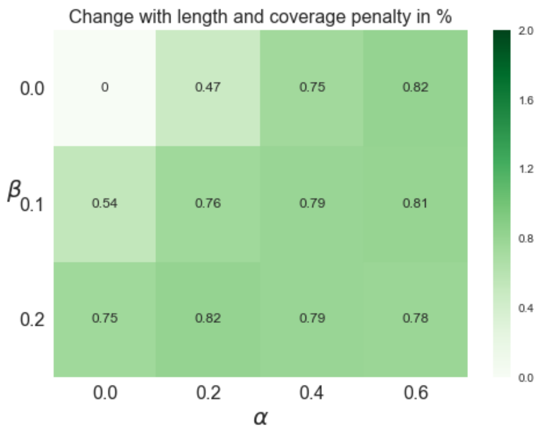

Figure 4: Relative change of performance (averaged over all 5 metrics) when Varying the α and β parameters for model (8).

els specializes in predicting a single token 100% of the time. Therefore, we conduct the E-Step on the the total probability of the target sequence instead. We illustrate in Figure 3 how this training regime works on S2S models with shared encoder parameters. Since each model is specialized on different sentence plans, averaging predictions of multiple models during inference, a technique commonly used with traditional ensembling approaches, does not lead to increased performance. Moreover, we confirm findings by Lee et al. (2017) who state that the models trained with sMCL overestimate their confidence when generating text. Therefore, choosing the model with the maximum likelihood prediction for a given input during inference does not lead to increased performance either. Since it is our goal to train a model that learns the best underlying template instead of generating diverse predictions, we instead generate text using only the model in the ensemble with the best perplexity on the validation set.

6 Experimental Details

For all experiments, we use a two-layer bidirectional encoder LSTM with d_{hid}=750, and d_{emb}=750. During training, we apply dropout with probability 0.2 and train models with Adam (Kingma and Ba, 2014) and an initial learning rate of 0.002. During experiments, we found Adam to significantly outperform stochastic gradient descent. We evaluate both *mlp* and *dot* attention types. The Transformer model is trained with 4 layers and d_{hid} and d_{emb}=512. We use the training rate schedule described by Vaswani et al. (2017), using Adam and a maximum learning rate of 0.1 after 2,000 warm-up steps. We apply the diverse ensembling technique to all approaches, training all models simultaneously for 4 epochs and then activating the sMCL loss. Finally, we show the change of a single model when activating different penalties. All models are implemented in OpenNMT-py (Klein et al., 2017)[1]. We note that due to its complexity, our re-implementation of the Transformer model still underperforms compared to the original paper. The parameters were found by grid search starting from the parameters used in the official baseline (Dušek and Jurčíček, 2016).

Since the test set was blind and used only for the submitted systems, we show results on the multi-reference validation set for the five metrics BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004), and CIDEr (Vedantam et al., 2015). Short descriptions of the metrics are shown in Table 2. We also report the validation perplexity, which does not correlate well with the other metrics.

7 Results

7.1 Validation Results

Table 3 shows results for different system configurations without penalties ($\alpha, \beta = 0.0$), using beam search with a beam size of 10 during inference. We compare our models to the official baseline that uses the TGEN (Dušek and Jurčíček, 2016) model. Except for the copy-only condition, *dot* outperforms *mlp*. Both copy-attention and diverse ensembling increase performance, and com-

[1]Code can be found at `https://github.com/sebastianGehrmann/OpenNMT-py/tree/diverse_ensemble`

#	Setup	PPL	BLEU	NIST	METEOR	ROUGE	CIDEr
(0)	Baseline	-	69.3	8.47	47.0	72.6	2.39
(1)	*mlp*	3.29	70.6	8.35	47.3	73.8	2.38
(2)	*dot*	3.31	71.1	8.43	47.4	73.7	2.35
(3)	*mlp*, copy	**3.13**	71.4	8.44	47.0	74.1	2.43
(4)	*dot*, copy	**3.13**	69.8	8.20	47.8	74.3	2.51
(5)	*mlp*, $K = 2$	3.27	72.6	8.70	48.5	74.8	2.52
(6)	*dot*, $K = 2$	3.14	73.3	8.68	**49.2**	**76.3**	2.61
(7)	*mlp*, copy, $K = 2$	3.22	73.6	8.74	48.5	75.5	**2.62**
(8)	*dot*, copy, $K = 2$	3.27	**74.3**	**8.76**	48.1	75.3	2.55
(9)	Transformer	3.22	69.0	8.22	47.8	74.9	2.45
(10)	Transformer, $K = 2$	3.22	73.7	8.75	48.9	**76.3**	2.56

Table 3: Results of different system configurations without length and coverage penalty.

MR	name[Wildwood] eatType[coffee shop] food[English] priceRange[moderate] customerRating[3 out of 5] near[Ranch]
(1)	Wildwood is a coffee shop providing English food in the moderate price range. It is located near Ranch.
(4)	Wildwood is a coffee shop providing English food in the moderate price range. It is near Ranch. Its customer rating is 3 out of 5.
(8).1	Wildwood is a moderately priced English coffee shop near Ranch. It has a customer rating of 3 out of 5.
(8).2	Wildwood is an English coffee shop near Ranch. It has a moderate price range and a customer rating of 3 out of 5.

Table 4: Examples of generated text by different systems. Numbers correspond to Table 3.

bining the two methods yields the highest BLEU and NIST scores across all conditions. Diverse ensembling also increases the performance with the Transformer model, which performs slightly worse than the standard model. Table 4 shows generated text from different models. We can observe that without copy, the model omits the rating, and that without ensembling, the sentence structure repeats and thus looks unnatural. With ensembling, both models produce sensible output. For the following results, we select model (8) from the table as the model configuration.

To investigate length and coverage penalties, we show the average relative change across all metrics while varying α and β in Figure 4. Both penalties increase average performance by up to 0.82%. Upon further investigation, we find that recall-based metrics (METEOR, ROUGE, CIDEr) increase while the precision-based metrics (BLEU, NIST) decrease when applying the penalty. We can explain this phenomenon with an increase in average length of the generated text by up to 2.4 words, which decreases the precision, but improves recall of the texts. Results for ensembling variations are shown in Table 5. While increasing K can lead to better template representations, every individual model will be trained on fewer data points. This can result in an increased generalization error. Therefore, we evaluate updating the top 2 models during the E step and setting K=3. While increasing K from 2 to 3 does not show a major increase in performance when updating only one model, the K=3 approach slightly outperforms the K=2 one with the top 2 updates.

Having the K models model completely disjoint data sets and use a disjoint set of parameters could be too strong of a separation. Therefore, we investigate the effect of sharing a subset of the parameters between individual models. Our results in rows (5)-(7) of Table 5 show only a minor improvement in recall-based approaches when sharing the word embeddings between models, but at the cost of a much lower BLEU and NIST score. Sharing more parameters further harms the model's performance. This is in line with our observations for other model configurations for this problem.

#	Setup	BLEU	NIST	METEOR	ROUGE	CIDEr
(1)	$K = 1$	69.8	8.20	47.8	74.3	2.51
(2)	$K = 2$	74.3	8.76	48.1	75.3	2.55
(3)	$K = 3$	73.6	8.73	48.8	75.5	2.64
(4)	$K = 3$, top 2	74.2	8.81	48.6	76.1	2.56
(5)	$K = 2$, share embedding	73.1	8.61	48.6	75.4	2.58
(6)	$K = 2$, share encoder	72.2	8.56	47.8	74.4	2.50
(7)	$K = 2$, share encoder + decoder	72.4	8.43	47.3	74.6	2.50

Table 5: The results of varying number of models and shared parameters while keeping setup (8) from Table 3

Setup	BLEU	NIST	METEOR	ROUGE	CIDEr
Main	65.0	8.53	43.9	68.7	2.09
Support 1	65.8	8.57 (8)	44.1	68.9 (9)	2.11
Support 2	66.2 (8)	8.60 (7)	**45.7 (1)**	70.4 (3)	**2.34 (1)**
Support 3	67.4 (3)	8.61 (6)	45.2 (4)	**70.8 (1)**	2.31 (3)

Table 6: The results of the submitted systems on the official test set. Notable rankings within the 60 submitted systems in parentheses.

7.2 Official Results

To evaluate results of our different approaches, we submitted the following four configurations of our system: **Main** *dot*, $K = 3$, top 2, no repeated sentence beginnings, **Support 1** *dot*, $K = 3$, top 2, **Support 2** Transformer, $K = 2$, **Support 3** *dot*, copy, $K = 2$

Since we observed that the models would often repeat the same sentence structure multiple times, we constrained the beam search in the main system to avoid beginning multiple sentences with the same bigram. While this modification leads to reduced scores on the automated metrics, it makes the text look more natural. We chose the Transformer model as another supporting system since it presents a drastically different approach from the other systems. All submitted models use diverse ensembling, since it improved performance across all configurations on the validation set. We used penalty parameters $\alpha = 0.4$ and $\beta = 0.1$ for all systems to focus on recall-based metrics. The official results (Dušek et al., 2018) shown in Table 6 confirm the validity of the approach, as our systems outperform competing systems in these; ranking first in ROUGE and CIDEr and sharing the first rank in METEOR.

The main system was also evaluated in a human evaluation by comparing its output to the other 18 primary systems. For a single meaning representation, crowd workers were asked to rank output from five systems at a time. Separate ranks were collected for the *quality* and *naturalness*. The results were then analyzed using the TrueSkill algorithm by Sakaguchi et al. (2014). The algorithm produced 5 clusters of systems for both quality and naturalness. Within clusters, no statistically significant difference between systems can be found. In both evaluations, our main system was placed into the second best cluster.

8 Conclusion

In this paper, we have shown three contributions towards the problem of natural language generation. We surveyed existing S2S modeling methods for their applicability to the NLG problem. We further showed that explicitly modeling different underlying structures in the data by applying the diverse ensembling technique can lead to a more robust learning process for structured, but noisy, data. Finally, we empirically evaluated the investigated methods and showed that a combination of them can lead to major improvements in multiple automatic evaluation metrics.

Acknowledgments

Sebastian, Falcon, and Alexander are supported by the Samsung Digital Media and Communications R&D Center. Work on OpenNMT is also supported by a gift from Systran.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*. pages 376–380.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pages 138–145.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491* .

Ondrej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *(in prep.)*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. 2012. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems*. pages 1799–1807.

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

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810* .

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

Kimin Lee, Changho Hwang, KyoungSoo Park, and Jinwoo Shin. 2017. Confident multiple choice learning. *arXiv preprint arXiv:1706.03475* .

Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. 2016. Stochastic multiple choice learning for training diverse deep ensembles. In *Advances in Neural Information Processing Systems*. pages 2119–2127.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computational Linguistics* .

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

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Saarbrücken, Germany. ArXiv:1706.09254. `https://arxiv.org/abs/1706.09254`.

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*. Association for Computational Linguistics, pages 27–32.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *WMT@ ACL*. pages 1–11.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers* .

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*. Association for Computational Linguistics, page 79.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 4566–4575.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

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. *arXiv preprint arXiv:1508.01745* .

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, 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. http://arxiv.org/abs/1609.08144.