Attending to Future Tokens for Bidirectional Sequence Generation

Carolin Lawrence and **Bhushan Kotnis** and **Mathias Niepert**
NEC Laboratories Europe
{carolin.lawrence, bhushan.kotnis, mathias.niepert}@neclab.eu

Abstract

Neural sequence generation is typically performed token-by-token and left-to-right. Whenever a token is generated only previously produced tokens are taken into consideration. In contrast, for problems such as sequence classification, *bidirectional* attention, which takes both past *and* future tokens into consideration, has been shown to perform much better. We propose to make the sequence generation process bidirectional by employing special placeholder tokens. Treated as a node in a fully connected graph, a placeholder token can take past and future tokens into consideration when generating the actual output token. We verify the effectiveness of our approach experimentally on two conversational tasks where the proposed bidirectional model outperforms competitive baselines by a large margin.

1 Introduction

When generating an output sequence, neural network models typically produce one token at a time. At each generation step, only the already produced sequence is taken into account. However, future and not-yet-produced tokens can also be highly relevant when choosing the current token. The importance of attending to both past and future tokens is apparent in self-attention architectures such as the Transformer (Vaswani et al., 2017). The self-attention module of a Transformer network treats a sequence *bidrectionally* as a fully connected graph of tokens – when a token is produced all other tokens are taken into consideration. However, this requires the entire sequence to be known a priori and when a Transformer is used for sequence generation, the self-attention process only includes previously produced tokens (Vaswani et al. (2017); Radford et al. (2019); *inter alia*). But the *bidirectional* self-attention is a crucial property of the highly successful language model BERT (Devlin et al., 2018). During the pre-training procedure of BERT, a fraction of input tokens is randomly masked out and the training objective is to predict these masked tokens correctly. BERT can then be fine-tuned for various classification tasks. Unfortunately, BERT cannot be directly used for sequence generation because the bidirectional nature of the approach requires the entire sequence to be known beforehand.

Inspired by BERT's masking-based objective, we propose to start out with a sequence of placeholder tokens which are iteratively replaced by tokens from the output vocabulary to eventually generate the full output sequence. For an example see Figure 1. With this novel model component, the self-attention of a Transformer can take both past and future tokens into consideration, leading to *Bi*directional *S*equence generati*on* (BiSon). Furthermore, it allows us to directly incorporate the pre-trained language model BERT and, to the best of our knowledge, for the first time directly fine-tune it for sequence generation.

BiSon makes two major contributions which we investigate in turn. First, we explore different stochastic placeholder replacement strategies to determine, at training time, where to position the placeholder tokens. This is crucial as we need the BiSon models to be exposed to a large number of heterogeneous placeholder configurations. Second, we explore several strategies for iteratively generating, at inference time, a complete output sequence from an initial sequence of placeholders.

We evaluate our bidirectional sequence generation approach on two conversational tasks. BiSon outperforms both competitive baselines and state of the art neural network approaches on both datasets by a significant margin.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1–10,
Hong Kong, China, November 3–7, 2019.

1	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$	$\bar{p}$
2	$\bar{p}$	you	been	$\bar{p}$	or	enrolled	in	an	$\bar{p}$	degree	or	program	?
3	Have	you	been	accepted	or	enrolled	in	an	accredited	degree	or	program	?

Figure 1: Example generation, going from a sequence of the placeholder token $\bar{p}$ (1), to an intermediate representation (2) and to the final output (3).

2 Sequence Generation with Transformers

For sequence-to-sequence tasks, an input sequence $x = x_1, x_2, \ldots, x_{|x|}$ is to be mapped to an output sequence $y = y_1, y_2, \ldots, y_{|y|}$ by some model π_θ with learnable parameters θ. For neural models, this is typically done by first encoding the input sequence x and then calling a decoder t times to produce a sequence y token-by-token, from left-to-right.

A popular choice for both encoder and decoder is the transformer (Vaswani et al., 2017). It takes a sequence of embedded tokens $s = s_1, s_2, \ldots, s_{|s|}$ and treats it as a fully connected graph over which a self-attention module is applied: for each token s_t in the sequence it assigns a probabilistic attention score a_t to every other token in the sentence. For the full mathematical details we refer the reader to (Vaswani et al., 2017).

Typically a transformer *encoder* is employed to encode x, whereas a transformer *decoder* is used to produce y. In contrast to the encoder, the decoder at time step t only has access to previously produced tokens $s_{<t} = s_1, s_2, \ldots, s_{t-1}$. Consequently, the attention module cannot take possible future tokens into account when making its decision at time t. Additionally, in this encoder-decoder framework, there is a disconnect between input x and output y because the self-attention modules are applied to x and y in isolation before they are combined.

The latter weakness has been overcome in recent work (Radford et al., 2018; Wolf et al., 2019; Radford et al., 2019) by feeding the concatenation $s = x \oplus y$ to a transformer decoder. At training time, given the current token s_t, the transformer is trained to predict the next word s_{t+1} via maximum likelihood estimation. At test time, the transformer is conditioned on x and then produces the output y token-by-token. But because the model is a transformer decoder, it is unable to take possible future tokens into account.

3 Bidirectional Sequence Generation

During sequence generation, we want to take both past and future tokens into account. More formally, at time t, we want to attend to both $s_1, \ldots, s_{t-1}$ as well as $s_{t+1}, \ldots, s_{|s|}$. To do this, we give the sequence $s = x \oplus y$, the concatenation of the sequences x and y, to a Transformer encoder, rather than a decoder. Of course, at inference time y is unknown. Thus, we propose to replace each token y_j with a *placeholder token* $\bar{p}$. Since the model needs to be exposed to heterogeneous placeholder token configurations during training time, we introduce a *placeholder strategy* that replaces some tokens y_j with placeholder tokens $\bar{p}$ at training time. Hence, during training, a sequence y is replaced by a sequence $p = p_1, p_2, \ldots, p_{|y|}$, where a token p_j is either the original token y_j or the placeholder token $\bar{p}$. We introduce two placeholder strategies in the following section. At inference time, p contains only placeholder tokens up to some pre-determined maximum sequence length.

With the placeholder strategy in place, a Transformer encoder is given the sequence $s = x \oplus p$ as input. The self-attention module then computes hidden representations r_t of each token s_t by attending to every other token in the sequence s. Because the output sequence is already present in the form of placeholder tokens both past tokens as well as future, not-yet-produced, tokens can be taken into consideration for every token s_t. Following the self-attention step, placeholder tokens are converted into a token from the output vocabulary with a language model (LM) classification layer, where for each placeholder p_t, its hidden representation r_t is mapped to a distribution d_t over the output vocabulary.

At training time, each output sequence token is fed the gold label and updates to the model π_θ are performed using stochastic gradient descent with a cross-entropy loss, i.e.

$$\mathcal{L}_{\pi_\theta} = -\frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{|y|}\log \pi_\theta(p_j = y_j|s),$$

where M is the size of a minibatch.

At inference time, the placeholder tokens can be replaced iteratively based on the probability distribution d_t over the output vocabulary for each placeholder p_t. Different sequence generation strategies are outlined in Section 3.2.

3.1 Placeholder Replacement Strategy

At inference time, the output sequence starts out with a sequence of placeholder tokens. To introduce this notion at training time, we require a strategy that replaces some output sequence tokens y_j with the placeholder token $\bar{p}$. The simplest approach would be to replace all output sequence tokens y_j with the placeholder token $\bar{p}$. However, with this approach the model is never confronted with a sequence containing a mix of output sequence tokens and placeholder tokens.

Due to the exponential number of possible replacement configurations per given token sequence, we introduce probabilistic generative models that we can use to draw diverse sequence replacements. To this end, we model the decision whether to use placeholder or input tokens with probabilistic models, two of which we propose in the following.

Bernoulli Random Variables (RV). We model each position of the input sequence with a binary random variable with a fixed mean. For every input sequence and every position i in the sequence, we draw from a Bernoulli variable with mean μ to decide whether to use y_i or $\bar{p}$. The expected number of placeholder tokens in a sequence of length $|y|$ is $|y|\mu$ and the variance is $\sigma^2 = |y|\mu(1-\mu)$. The variance of this strategy is determined by the mean and, therefore, the probabilistic model has one tunable parameter μ.

Gaussian Random Variables. The *number of placeholder tokens* of the input sequence p can be seen as drawn from an unknown optimal Binomial distribution. We can approximate this Binomial with a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the mean and σ the standard deviation and they are considered hyperparameters. More formally, for every input sequence, we draw a value $P \sim \mathcal{N}(\mu, \sigma^2)$. Multiplied with the sequence length $|y|$, the nearest integer value $\lfloor(|y| \cdot P)\rceil$ is used as the number of placeholder tokens for the given sequence. The positions of the placeholder tokens in the sequence are then determined at random. The resulting probabilistic model's two parameters (mean μ and standard deviation σ) are treated as hyperparameters and are not updated during training. Being able to tune the variance parameter independently from the mean parameter might provide an advantage over the parameterization with Bernoulli RVs.

3.2 Sequence Generation Strategies

Starting with a sequence of placeholder tokens at inference time, it is possible to generate output token sequences in arbitrary order. We experiment with the following strategies. The distribution in all of these strategies are the distributions d_t $(t = 1, ..., n)$ for the placeholders over the output vocabulary. We use the term *uncover* to mean that an output token is generated for a placeholder token.

One-step greedy. In a single time step, all placeholder tokens are uncovered simultaneously by picking the most probable token from the output vocabulary for each placeholder.

Highest probability. Placeholders are replaced iteratively and the placeholder to be uncovered is the placeholder that assigns the highest probability to a token from the output vocabulary, indicating the model is the most sure about this token.

Lowest entropy. Placeholders are replaced iteratively and the placeholder to be uncovered is the placeholder that exhibits the lowest entropy over its output vocabulary distribution and the most likely token at this position is chosen. Intuitively, the lowest entropy indicates the position where the uncertainty of the model to decide between tokens of the output vocabulary is the lowest.

Left-to-right. Placeholders are replaced iteratively, moving from left-to-right and thus mimicking the typical writing style for English. Note that this approach still differs from the Transformer decoders because future tokens are considered via the placeholder representations.

No look ahead. To test whether future placeholders hold useful information, we consider an adversarial sequence generation strategy: Again we iteratively uncover placeholders from left-to-right, but we suppress all attention flows from future placeholders. This imitates the behaviour of a transformer decoder but follows the idea of predicting a token on a placeholder, rather than predicting the next word as is typically done in transformer decoders. If this performs worse than left-to-right, there is indeed valuable information in future placeholder tokens.

4 Experiments

We conduct a series of experiments to explore BiSon's behavior. First, we want to compare two token replacement strategies for training as well as the four generation strategies for inference. Second, we want to compare BiSon to state of the art methods and investigate the impact of its ability to attend to future tokens.

4.1 Datasets

We run experiments on the two following conversational datasets.

Goal-oriented ShARC (Saeidi et al., 2018). ShARC is a dialogue, text-based question-answering dataset. Unlike many popular QA datasets, answers cannot simply be extracted from the text. Given a regulatory text, such as a text from the UK government's website, and a user scenario with corresponding question, it is necessary to interpret the text in the context of the specific user's needs. Before generating its final answer, a system may generate clarification questions. Finally, the system decides if the answer to the user's original question is "*Yes*", "*No*" or "*Irrelevant*" where the latter means the question cannot be answered with the given text.

We perform the evaluation with the official ShARC script. For a set of generated clarification questions, it computes BLEU n-gram scores for $n = 1, 2, 3, 4$ using a set of clarification question in the set of gold responses. In each step of the conversation, the model under evaluation generates an output token sequence. This output is automatically assigned to the category "*More*" if it is a clarification question, and to "*Yes*", "*No*", and "*Irrelevant*" otherwise. Since this is a classification task we can compute micro and macro accuracy for it. The final model is chosen using the highest BLEU-4 score on the development set.

The ShARC dataset has a hidden test set and, therefore, it is not feasible to evaluate our various model variants. Hence, we take 30 unique rule texts and their corresponding training examples from the training set. This leads to a new development set of 2,465 instances and leaves the official development set to be used as a test set here. Finally we submitted our best model to be evaluated on the hidden test set.

Free-form Daily Dialog (Li et al., 2017). Daily Dialog is a dataset of written conversations occurring in daily life. Following the authors of the corpus, we report BLEU n-gram scores for $n = 1, 2, 3, 4$ for the generated output sequences with respect to the given gold responses. We tokenize these responses equally to ensure a fair comparison.

4.2 BiSon Settings

We implement BiSon based on the BERT Pytorch code[1] and initialize with the pre-trained BERT model BERT-BASE-UNCASED (Devlin et al., 2018). Consequently we employ the same model architecture and tokenisation as (Devlin et al., 2018) resulting in a model with about 110M parameters. To remain compatible with the BERT model, we prepend each sequence with a [CLS] token and place a [SEP] token after the input context. Similarly, producing a second [SEP] token indicates the end of sequence generation. For input context of ShARC, we follow Saeidi et al. (2018) and use the concatenation of question, rule text, scenario and history. The input context for Daily Dialog is the concatenation of all previous utterances.

On the ShARC and Daily Dialog training sets we train for 20 and 40 epochs, respectively, which equates in each case to about $200k$ seen examples. As optimizer we used Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a L2 weight decay of 0.01 and a learning rate warm-up over the first 10% of training steps. As learning rates we consider both the pre-training learning rate of BERT $1e$-4 and the fine-tuning learning rate $3e$-5. On preliminary experiments $3e$-5 proved to be best for ShARC, whereas it is $1e$-4 for Daily Dialog. We set the batch size to 15. Finally, the maximum sequence generation length, is set to 50 for ShARC and to 100 for Daily Dialog, which was chosen based on values observed in the training data. As the maximum sequence length of the BERT model is 512, longer input sequences are truncated accordingly. For the main results, we employ the sequence generation strategy left-

[1] https://github.com/huggingface/pytorch-pretrained-BERT

to-right, which we found to work best. Later on we also report results for the other strategies.

For the Bernoulli RV approach, we test $\mu \in [0.2, 0.8]$ with increments of 0.1. For the Gaussian RV approach, we test all possible combinations for the following hyperparmeters $\mu = \{0.4, 0.5, 0.6\}$ and $\sigma = \{0.3, 0.6, 0.9\}$. The best combination on the SHARC dev set is $\mu = 0.5, \sigma = 0.6$. It outperforms the best Bernoulli approach ($\mu = 0.7$) by 3.4 point in BLEU-4 score. Some Bernoulli experiments in fact only produced a very small number of clarification question, e.g. $\mu = 0.5$ only generated 9 clarification questions on the development set, whereas in the ground truth responses 846 clarification questions occur. This suggests that a high variance is important, as the Bernoulli setups all have a variance of 0.25 or lower and our best Gaussian approach has a variance of 0.6. We directly employ the Gaussian distribution with $\mu = 0.5, \sigma = 0.6$ on the DAILY DIALOG task.

4.3 Baselines

To measure the success of our proposed approach, we consider the following three baselines.

Encoder-Decoder Transformer (E&D). First, we compare our bidirectional encoder to a standard encoder-decoder Transformer where the decoder only has access to tokens produced so far to compute its self-attention. We use the implementation of OpenNMT (Klein et al., 2017) and employ the parameters suggested by them, but adjust the learning rate to 0.1, which we found to work better for both datasets. Additionally, we increased the word and hidden dimension size to 768 and the number of attention heads to 12 to match the capacity of our model. Training ran for 50 epochs. Needing both an encoder and a decoder, this leads to a total of about 270M parameters.

Encoder-Decoder Transformer with BERT (E&D+B). The power of our bidirectional decoder stems from two advantages. First, we can initialize our model with the pre-trained BERT-BASE-UNCASED model. Second, the decoding process is bidirectional. It would be possible to transfer the first advantage to an encoder-decoder framework by using BERT embeddings. This is however only possible for the input sequence, because the bidirectionality of BERT requires the entire sequence to be available beforehand. Thus, we modify implementation of OpenNMT to use the BERT model as the encoder. The weights are frozen

	Model	Micro Acc.	Macro Acc.	B-1	B-4
SHARC	E&D	31.9	38.9	17.1	1.9
SHARC	E&D+B	54.7	60.4	24.3	4.3
SHARC	GPT2	60.4	65.1	53.7	33.9
SHARC	BISON	**64.9**	**68.8**	**61.8**	**46.2**

Table 1: Results on the SHARC test set, averaged over 3 independent runs for GPT2 and BISON, reporting micro accuracy and macro accuracy in terms of the classification task and BLEU-1 and BLEU-4 on instances for which a clarification question was generated. E&D uses no language model pre-training.

	Model	Micro Acc.	Macro Acc.	B-1	B-4
SHARC	E3	**67.6**	**73.3**	54.1	38.7
SHARC	BISON	66.9	71.6	**58.8**	**44.3**

Table 2: Results on the official hidden SHARC test set of our model compared to the best model on the leaderboard, E3 (Zhong and Zettlemoyer, 2019).

when training the decoder, which produced better results than allowing the gradients to also flow through the BERT model. Again, with both an encoder and decoder, this leads to a total of about 270M parameters.

GPT2. Radford et al. (2019) present a transformer decoder, GPT2, trained as a language model on large amounts of monolingual text. Radford et al. (2019) showed that it is possible to perform various tasks in a zero-shot setting by priming the language model with an input and letting it generate further words greedily. This setup can be transferred to a supervised setting, where the model is fine-tuned to a dataset by using maximum likelihood estimation to increase the probability of the gold output sequence (Wolf et al., 2019). As the starting point for the supervised learning, we initialize the model with the pre-trained model GPT-2-117M released by Radford et al. (2019)[2] and then fine-tune. With 117M parameters, this model is comparable to our model. Unlike baseline 2, this setup can directly employ a pre-trained model as our approach can, but it is not bidirectional.

[2] https://github.com/openai/gpt-2

4.4 Results

We report the results of our approach, the various baselines, as well as the previous state-of-the-art (SOTA) scores where applicable in Table 1 and 2 for SHARC and in Table 3 for DAILY DIALOG.

On the SHARC dataset, we observe very poor BLEU-4 performance for the encoder-decoder Transformer (E&D), which is consistent with results from Saeidi et al. (2018), who could not get a LSTM-based network to work without an additional classification head. Adding BERT (E&D+B) slightly improves performance. By directly leveraging a pre-trained model, GPT2 outperforms the previous models by a large margin, reaching 33.9% on BLEU-4 and a micro accuracy of 60.4%. BISON is able to take future tokens into consideration and outperforms GPT2 by 12.3 percentage points in BLEU-4 and by 4.5 points in micro accuracy.

We submitted the best BISON model out of the random three of Table 1 to be evaluated on the hidden test set and report results in comparison to the best model on the leaderboard,[3] E3 (Zhong and Zettlemoyer, 2019) in Table 2. BISON outperforms E3 by 5.6 BLEU-4 points, while it is only slightly worse than E3 in terms of accuracy.

On the DAILY DIALOG dataset the information retrieval-based method (IR in Table 3) introduced by Li et al. (2017) is very strong and outperforms the best end-to-end model (E2E) (Luo et al., 2018) by over 16 percentage points in BLEU-4. The best end-to-end model is based on LSTMs and Luo et al. (2018) report performance increases when adding an attention module to their setup. The encoder-decoder transformer (E&D) outperforms this setup by over 2 percentage points in BLEU-4 and we conjecture that this is due to the transformer making more effective use of the attention principle. Adding BERT (E&D+B) does not help much for this dataset. But again we observe a large increase of performance when directly employing pre-trained models. GPT2 performs on par with the IR SOTA, achieving a BLEU-4 score of 19.4%. Again, BISON can outperform GPT2, here with a difference of 6.2 points in BLEU-4 and even larger increases in the other scores.

Effect of bidirectionality. To investigate that our model benefits from bidirectionality, we consider a setup where BISON isn't allowed to attend to future tokens during prediction (see Table 4). It causes a drop in BLEU-4 performance of about 25 points on the SHARC dataset and a drop of 10 points on the DAILY DIALOG dataset. This showcases that BISON during training has learnt to rely on the ability to attend to future tokens.

Effect of pre-trained model. We are curious how big the effect of the pre-trained model is. Thus, instead of starting with the BERT-BASE-UNCASED weights, we initialize BISON with random weights drawn from a normal distribution with mean 0.0 and standard deviation of 0.02. Results are presented in Table 5 for SHARC and DAILY DIALOG. Even without a pre-trained language model, our approach can outperform the standard encoder-decoder transformer framework (E&D) on both datasets, although we had to increase the number of epochs for the SHARC dataset to 40. On the DAILY DIALOG task, we are even able to outperform GPT2. This demonstrates

	Model	B-1	B-2	B-3	B-4
DAILY DIALOG	IR	-	25.8	20.4	19.4
	E2E	14.2	5.7	3.8	2.8
	E&D	22.3	6.8	5.7	5.2
	E&D+B	26.1	7.3	6.0	5.5
	GPT2	42.3	23.6	20.7	19.4
	BISON	**54.9**	**32.6**	**28.0**	**25.6**

Table 3: BLEU n-gram scores for $n = 1, 2, 3, 4$ on the DailyDialog test set, averaged over 3 independent runs for GPT2 and BISON. Models before the line do not make use of a pre-trained language model. IR (SOTA) (Li et al., 2017) and E2E (SOTA) (Luo et al., 2018) are, to the best of our knowledge, the best previously published scores for information retrieval and end-to-end approaches.

	Model	Micro Acc.	Macro Acc.	B-1	B-4
SHARC	BISON	**64.9**	**68.8**	**61.8**	**46.2**
	past only	64.3	67.4	35.0	21.3
	Model	B-1	B-2	B-3	B-4
DD	BISON	**54.9**	**32.6**	**28.0**	**25.6**
	past only	48.0	24.6	18.5	14.8

Table 4: Comparison of BISON to a setup where BISON isn't allowed to attend to future tokens, i.e. past only, for SHARC and DAILY DIALOG (DD).

[3] https://sharc-data.github.io/leaderboard.html, 19 August 2019

	Model	Micro Acc.	Macro Acc.	B-1	B-4
SHARC	E&D	31.9	38.9	17.1	1.9
	BISON	**52.9**	**57.4**	**21.9**	**2.3**
	Model	B-1	B-2	B-3	B-4
DD	E&D	22.3	6.8	5.7	5.2
	BISON	**46.3**	**27.0**	**23.6**	**22.4**

Table 5: Best end-to-end models that do not use a pre-trained language model in comparison with BISON that uses randomly initialized weights for SHARC and DAILY DIALOG (DD), averaged over 3 runs.

Strategy	SHARC	DAILY DIALOG
one step greedy	22.9	9.3
lowest entropy	40.3	16.8
highest probability	**50.9**	16.4
left-to-right	46.2	**23.8**

Table 6: BLEU-4 using various sequence generation strategies for BISON on SHARC and DAILY DIALOG.

the effectiveness of our approach in itself, free of any pre-trained language model.

Effect of sequence generation strategies. We present the different sequence generation strategies in Table 6. The best overall sequence generation strategy is to predict from left to right which achieves good results on both datasets. On the SHARC dataset the highest probability approach performs better than left-to-right. However, on DAILY DIALOG this approach is not as successful. This suggests that it might be worth selecting the best sequence generation strategy for each dataset individually. However, we hypothesize that left-to-right works consistently well due to the left-to-right nature of the English language. A brief experiment with a right-to-left strategy gave poor results.

5 Analysis

We believe that the placeholders capture sequential information present in the language model learned during pre-training. After running a transformer encoder where each position can attend to every other position, a placeholder token will have a probability distribution over the output vocabulary and this distribution is informed by all other tokens in input and output. Thus, a placeholder could be seen as a mixture of tokens with varying probabilities. As placeholders are subsequently uncovered, the other placeholders can update their distribution by taking the newly revealed token into consideration.

Dataset	α^1	α^2	α^3
SHARC	92.6±3.1	5.2±2.4	2.2±1.8
DD	97.0±2.5	2.3±2.3	0.7±0.4
		$\bar{\alpha}^2$	$\bar{\alpha}^3$
SHARC	-	71.7±13.7	28.3±13.7
DD	-	70.2±14.5	29.8±14.5

Table 7: Average attention weights and standard deviation when predicting from left-to-right on both SHARC and DAILY DIALOG (DD) for different parts of the sequence, where α^1 is for the input sequence x, $\alpha^2/\bar{\alpha}^2$ is for the already produced sequence y and $\alpha^3/\bar{\alpha}^3$ is for the sequence of remaining placeholder tokens p. α^k are the normalized attention weights across all three parts, whereas $\bar{\alpha}^k$ normalizes over the second and third part.

For example, in Figure 2, for the sentence“is the animal an endangered animal ?”, while generating “endangered”, the self-attention head pays attention to the next placeholder token, which in the next step is revealed to be “animal”. While producing “endangered”, the distribution for the next position already placed a high probability on “animal”, thus the current token can take this into consideration and produces “endangered”. Further heat maps demonstrating this can be found in the appendix.

The quantify this intuition, we measure the average attention score on various parts of the sequence. For this, we use the left-to-right prediction strategy. Thus, at time t, we can decompose our sequence into three parts: $s = x \oplus y \oplus p$, where x is the input, y the already produced sequence and p the remaining sequence of placeholder tokens. For each attention head, we can decompose the attention probabilities into the three parts,

1. attention on the input text,
2. attention on the current word and already generated words (left of the current word),
3. attention on words that are yet to be generated (right of the current word).

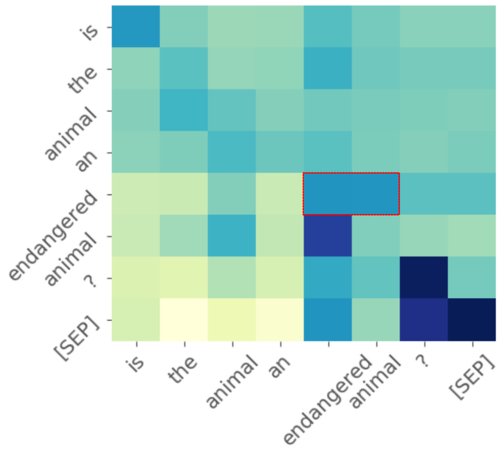

Figure 2: Heat map (darker hues indicate higher attention) that shows an example of where an attention head looks into the future while generating from left to right. Each row shows the attention over the output sequence for this row's placeholder token at that point in time. Word in previous rows have been produced already, whereas words of later rows still hold placeholder tokens. Thus the upper triangle of the matrix shows the attention that is paid to future tokens. The red square shows that while generating the token "endangered", the attention head already takes the next placeholder into account, which is revealed to be "animal" in the next step. Best viewed in color.

This is mathematically expressed as

$$a_t = a_{0:|x|} \oplus a_{|x|+1:|x|+t} \oplus a_{|x|+t+1,|s|},$$

where $|s|$ is the maximum possible sequence length. For each part we can calculate an average leading to three values, a_t^1, a_t^2 and a_t^3.

Averaged over all T generation time steps and all D data points, we can derive a score for each part k, $k = 1, 2, 3$ and each attention head h:

$$\alpha_h^k = \frac{1}{D}\frac{1}{T}\sum_{d=1}^{D}\sum_{t=1}^{T} a_{d,t}^k$$

Note that we use the attention heads in the final BERT self-attention layer. Averaging over all H attention heads, $\alpha^k = \frac{1}{H}\sum_{h=1}^{H}\alpha_h^k$, leads to the results reported in Table 7 for both datasets. Unsurprisingly, we find that with scores of over 90% for both datasets the majority of the attention is focused on the first part, i.e. the conditioning input x (see α^1 in Table 7). The remaining attention is split between the already produced sequence (α^2) and the future tokens (α^3).

To directly compare the relationship within the sequence generation, we re-normalize over α^2 and α^3, leading to new values $\bar{\alpha}^2$ and $\bar{\alpha}^3$ (see Table 7). Here we can see that the past, already produced tokens are about twice as important as the future, not-yet-produced tokens. But with scores of just under 30% on both datasets, we see that a substantial amount of attention is also focused on the future, not-yet-produced tokens.

Interestingly, with a standard deviation of about 14%, the values of $\bar{\alpha}^2$ and $\bar{\alpha}^3$ vary strongly across the different attention heads. For example on the SHARC dataset, we find one attention head where only about 9% is focused on the future and another where it is about 64% and thus this attention head pays more attention to the future than the past. A graphical overview can be found in the appendix for both datasets.

6 Related Work

Transformers (Vaswani et al., 2017) model sequences as fully connected graphs and apply a bidirectional self-attention module where every token can attend to every other token. Because of this a Transformer is not restricted to sequential orderings. However, Vaswani et al. (2017); *inter alia* still restrict themselves to producing tokens from left-to-right and only allow a Transformer decoder to attend to previously produced tokens. Recently, several attempts have been made to lift the left-to-right restriction in Transformer or LSTM-based models (Gu et al., 2019; Stern et al., 2019; Welleck et al., 2019; Zhou et al., 2019), but in those approaches it is not possible to attend to future, not-yet-produced tokens.

Concurrently to our work, (Ghazvininejad et al., 2019) proposed a similar placeholder strategy approach for generating in the context of machine translation. However, they employ an encoder-decoder framework, whereas we only require an encoder, which more closely links input and output via a single shared attention module. Furthermore, they only consider uniform sampling of placeholders whereas we found that the higher variance, which we can control with the Gaussian random variable approach, leads to better results.

Bidirectionality is one of the crucial ingredients in the success of the recently proposed unsupervised language model BERT (Devlin et al., 2018). For this, Devlin et al. (2018) propose a Transformer *encoder* to take full advantage of the bidirectional nature of the Transformer. Their resulting model, BERT, can directly be applied to various classification tasks but not to sequence generation tasks. Our approach shows how a Transformer encoder can be used for sequence generation and this allows us to directly incorporate BERT into our experiments.

GPT (Radford et al., 2018) and GPT2 (Radford et al., 2019) are both pre-trained language models that use a Transformer *decoder* instead, which can only attend to already produced tokens. For dialogue, the GPT model has been fine-tuned for the chit-chat dataset PersonaChat (Zhang et al., 2018) by Wolf et al. (2019). While GPT and GPT2 can immediately be used as a sequence generators, these models do not offer bidirectionality and they cannot attend to not-yet-produced tokens. Our bidirectional encoder for sequence generation can combine the best of both worlds.

7 Conclusion

We introduced bidirectional sequence generation by employing placeholders in the output sequence. These placeholder tokens are subsequently replaced by tokens of the output vocabulary. Crucially, this allows a transformer encoder to attend to both past and future, not-yet-produced token. Simply masking all placeholder tokens is not feasible. Instead we investigated two placeholder strategies, based on Bernoulli and Gaussian random variables. At prediction time, our approach is not restricted to produce the output sequence from left to right. However, this strategy proved to produce most consistent results in our experiments.

Our approach outperforms previous end-to-end approaches that do not make use of any pre-trained language models. In conjunction with the pre-trained language model BERT, our bidirectional sequence generation approach allows us to achieve new state-of-art results on both conversational tasks. In the future, we would like to apply our approach to other sequence generation tasks. Additionally, we wonder if a further performance increase could be achieved if the pre-training of BERT would employ our placeholder strategy.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *ArXiv e-prints*, 1810.04805.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-Time Machine Translation with Conditional Masked Language Models. *arXiv:1904.09324 [cs, stat]*. ArXiv: 1904.09324.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *CoRR*, abs/1902.01370.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*, 1701.02810.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP)*, Taipei, Taiwan.

Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng, and Xu Sun. 2018. An auto-encoder matching model for learning utterance-level semantic dependency in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Technical Report Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, and Dario Amodei. 2019. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *CoRR*, abs/1902.03249.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NIPS)*.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. *CoRR*, abs/1902.02192.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. *ArXiv e-prints*, 1901.08149.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Victor Zhong and Luke Zettlemoyer. 2019. E3: Entailment-driven extracting and editing for conversational machine reading. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.