KERMIT: Generative Insertion-Based Modeling for Sequences

William Chan*[1], **Nikita Kitaev***[1,3], **Kelvin Guu***[2], **Mitchell Stern***[1,3], **Jakob Uszkoreit**[1]
[1]Google Research, Brain Team
[2]Google Research, AI Language Team
[3]University of California, Berkeley
`{williamchan,kguu,usz}@google.com`
`{kitaev,mitchell}@berkeley.edu`

Abstract

We present KERMIT, a simple insertion-based approach to generative modeling for sequences and sequence pairs. KERMIT models the joint distribution and its decompositions (i.e., marginals and conditionals) using a single neural network and, unlike much prior work, does not rely on a prespecified factorization of the data distribution. During training, one can feed KERMIT paired data (x, y) to learn the joint distribution $p(x, y)$, and optionally mix in unpaired data x or y to refine the marginals $p(x)$ or $p(y)$. During inference, we have access to the conditionals $p(x \mid y)$ and $p(y \mid x)$ in both directions. We can also sample from the joint distribution or the marginals. The model supports both serial fully autoregressive decoding and parallel partially autoregressive decoding, with the latter exhibiting an empirically logarithmic runtime. We demonstrate through experiments in machine translation, representation learning, and zero-shot cloze question answering that our unified approach is capable of matching or exceeding the performance of dedicated state-of-the-art systems across a wide range of tasks without the need for problem-specific architectural adaptation.

1 Introduction

Neural sequence models (Sutskever et al., 2014; Cho et al., 2014) have been successfully applied to many conditional generation applications, including machine translation (Bahdanau et al., 2015; Luong et al., 2015), speech recognition (Chan et al., 2016; Bahdanau et al., 2016), speech synthesis (Oord et al., 2016; Wang et al., 2017) and image captioning (Vinyals et al., 2015; Xu et al., 2015). Much of the prior work in this area follows the seq2seq encoder-decoder paradigm, where an encoder builds a representation of an observed sequence x, and a decoder gives the conditional output distribution $p(y \mid x)$ according to a predetermined factorization, usually left-to-right.

While effective for straightforward conditional generation, such an approach is inflexible and cannot readily be applied to other inference tasks such as non-left-to-right generation or infilling. In this work, we present a more general approach called Kontextuell Encoder Representations Made by Insertion Transformations, or KERMIT for short. KERMIT is a simple architecture that directly models the joint distribution $p(x, y)$ and its decompositions (such as the marginals $p(x)$ and $p(y)$

*Equal contribution. WC initiated the KERMIT project for machine translation, implemented the corresponding code and experiments (Section 4.1) and advised the project. NK proposed using the same model for text generation, implemented and evaluated different monolingual pre-training approaches, and conducted all representation learning experiments (Section 4.2). KG proposed using KERMIT as a zero-shot QA model and conducted all associated experiments (Section 4.3); he also co-developed KERMIT's training and inference infrastructure. MS developed the mathematical formalism for the model (Section 3) and assisted in the implementation of KERMIT for translation. JU helped conceive the initial idea and advised the project.

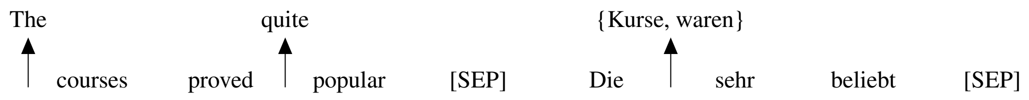

Figure 1: An example of the KERMIT insertion objective for the English ↔ German translation pair "The courses proved quite popular" ↔ "Die Kurse waren sehr beliebt". The model is trained to predict the set of words that need to be inserted at each location. By incurring a loss on both sides, our system learns a fully generative model of the joint distribution over (x, y) pairs, and can accommodate arbitrary generation orders.

	Machine Translation En → De (BLEU)	**Representation Learning** GLUE	**Cloze Question Answering** Zero-shot SQuAD (F1)
Autoregressive (Transformer, GPT, GPT-2)	27.3[a]	72.8[b]	16.6
Masking (BERT)	N/A	80.5[c]	18.9
Insertion (KERMIT – Our Work)	27.8	79.8	30.3

Table 1: The KERMIT architecture works well for three categories of tasks: machine translation, representation learning, and zero-shot cloze question answering. [a]Vaswani et al. (2017) [b]Radford et al. (2018) [c]Devlin et al. (2019)

and the conditionals $p(y \mid x)$ and $p(x \mid y)$) in a unified manner. In contrast with traditional seq2seq models, KERMIT does not rely on a prespecified factorization, but is instead able to condition on whatever information is available and infer what remains.

During training, we present KERMIT with paired data (x, y) to learn the joint, and can optionally mix in unpaired data x or y to refine the marginals in a semi-supervised setting. At test time, a single KERMIT model can be used for conditional inference in either direction by restricting the output distribution to $p(x \mid y)$ or $p(y \mid x)$ as required. We can also generate paired samples from the joint distribution $(x, y) \sim p(x, y)$, or unpaired samples from the marginals $x \sim p(x)$ or $y \sim p(y)$.

KERMIT uses a simple architecture and is easy to implement. It does not have a separate encoder and decoder, nor does it require causality masks. In our implementation, KERMIT consists of a single Transformer decoder stack (Vaswani et al., 2017). The model is trained to insert the missing tokens into any partially-complete sequence, as shown in Figure 1. We describe the implementation in more detail in Section 3.

We apply KERMIT to a diverse set of tasks, finding that our unified approach is capable of matching or exceeding the performance of dedicated state-of-the-art systems without the need for problem-specific components. We first apply KERMIT to machine translation, where the inputs and outputs are parallel sentence pairs. Then, like its friends ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and ERNIE (Sun et al., 2019), we can also use KERMIT for self-supervised representation learning for use in downstream NLP tasks. Finally, we apply KERMIT to a zero-shot cloze question-answering task demonstrating the infilling capabilities of the model. Table 1 summarizes our results on all three tasks compared to other highly tuned models: Transformer, BERT, GPT and GPT-2.

2 Background

In this section, we define some notation and give a brief review of existing sequence models, including autoregressive left-to-right models (Sutskever et al., 2014; Cho et al., 2014) and masked language models (Devlin et al., 2019).

2.1 Autoregressive Left-to-Right Models

Let $\mathcal{X}$ and $\mathcal{Y}$ be the set of all input and output sequences, respectively. In a standard sequence-to-sequence task, we are presented with training data consisting of sequence pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, e.g. parallel translations, and we aim to learn the conditional distribution $p(y \mid x)$. Traditional autoregressive models (Sutskever et al., 2014; Cho et al., 2014) use a left-to-right factorization, decomposing the distribution as a chain of predictions conditioning on the input x and prefixes $y_{<t}$:

$$p(y \mid x) = \prod_t p(y_t \mid x, y_{<t}). \tag{1}$$

This structure is also used for unconditional sequence tasks such as language modeling where the goal is to learn an unconditional output distribution on its own. A left-to-right factorization is convenient because it allows for exact log-likelihood computation, thereby permitting efficient maximum likelihood estimation. It also leads to simple approximate inference algorithms such as greedy decoding

$$\hat{y}_t = \operatorname*{argmax}_y p(y \mid x, \hat{y}_{<t}) \tag{2}$$

or beam search over sets of multiple hypotheses.

However, there are some drawbacks to the autoregressive approach. First, in the case of conditional generation, it cannot handle situations where the input x is only partially observed. Second, since it utilizes a fixed left-to-right factorization, it cannot be used for other inference tasks like infilling where generation is not monotonic. Moreover, standard inference algorithms require n generation steps to generate n tokens, which could be a bottleneck in end-use applications.

2.2 Masked Language Models

Masked Language Models (MLMs) (Devlin et al., 2019) comprise another class of models targeting the unconditional setting. For MLMs, a partial canvas $x_s \subseteq x$ is observed where some of the tokens in x have been masked out, and the objective is to recover x from x_s. For example, for a ground truth canvas $x^* = (A, B, C, D, E)$ and a partial canvas $x^*_s = (A, \_, C, D, \_)$, the model should learn to replace the second blank with B and the last blank with E. The model outputs an independent prediction at each position, and its objective is to maximize $p(x \mid x_s)$.

Because the exact locations of the slots are known in x_s, the model does not need to predict where the missing items are located, but only what they should be. Consequently, the model is not immediately suitable for generation, as the canvas size needs to be fixed during inference and cannot change over time (i.e., $|x_s| = |x|$). MLMs have been successfully applied in self-supervised representation learning settings, leading to strong results on downstream language tasks (Devlin et al., 2019).

3 KERMIT

In this section we propose KERMIT, a novel insertion-based generative model. Unlike the prior work mentioned in Section 2, KERMIT does not have the rigid construction of modeling the target sequence given some fully observed source sequence, nor does it assume a left-to-right factorization (and generation order) of the output sequence. To motivate and arrive at our model, we formalize then extend a recent insertion-based conditional modeling framework proposed by Stern et al. (2019).

We begin with the unconditional setting. In order to model sequences without requiring a fixed factorization or imposing constraints on the order of generation, we make use of a framework in which sequences are constructed via insertion operations. Given a sequence $x = (x_1, \ldots, x_n)$ and a generation order z represented as a permutation of the indices $\{1, \ldots, n\}$, we define the corresponding sequence $((c^z_1, l^z_1), \ldots, (c^z_n, l^z_n))$ of insertion operations which produces x according to order z. Here, $c^z_i \in \mathcal{C}$ is an element of the vocabulary and $1 \leq l^z_i \leq i$ is an insertion location relative to the current hypothesis. For example, if constructing the sequence (A, B, C) as $() \rightarrow (C) \rightarrow (A, C) \rightarrow (A, B, C)$, we would have $z = (3, 1, 2)$ with $(c^z_1, l^z_1) = (C, 1), (c^z_2, l^z_2) = (A, 1), (c^z_3, l^z_3) = (B, 2)$.

Next let $(x^{z,i}_1, \ldots, x^{z,i}_i)$ denote the subsequence of x corresponding to the (ordered) extraction of the elements at indices $\{z_1, \ldots, z_i\}$. This is the partial output at iteration i. Note that this will be the same for all permutations z with the same unordered set of indices in the first i positions. For the example above for instance, we have $(x^{z,2}_1, x^{z,2}_2) = (A, C)$.

Armed with these definitions, we can now write out $p(x)$ as a marginalization over all possible orders $z \in S_n$ for sequence length n, where S_n denotes the set of all permutations on n elements:

$$p(x) = \sum_{z \in S_n} p(x, z) \tag{3}$$

$$= \sum_{z \in S_n} p(z) p(x \mid z) \tag{4}$$

$$= \sum_{z \in S_n} p(z) \prod_{i=1}^{n} p((c_i^z, l_i^z) \mid (c_1^z, l_1^z), \ldots, (c_{i-1}^z, l_{i-1}^z)) \tag{5}$$

$$= \sum_{z \in S_n} p(z) \prod_{i=1}^{n} p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}), \tag{6}$$

where the last line encodes the Markov assumption that the order of insertions leading to a given canvas is not important, just the result. Typically we will use a uniform prior over permutations for $p(z)$, though other options are available, such as the balanced binary tree prior described by Stern et al. (2019).

3.1 Learning

Although exact computation of the log-likelihood is intractable due to the marginalization over the generation order z, we can lower bound the log-likelihood using Jensen's inequality via

$$\log p(x) = \log \sum_{z \in S_n} p(z) p(x \mid z) \tag{7}$$

$$\geq \sum_{z \in S_n} p(z) \log p(x \mid z) \quad =: \mathcal{L}(x). \tag{8}$$

Substituting in our expression for $p(x \mid z)$ from above, we have

$$\mathcal{L}(x) = \sum_{z \in S_n} p(z) \log \prod_{i=1}^{n} p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}) \tag{9}$$

$$= \sum_{z \in S_n} p(z) \sum_{i=1}^{n} \log p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}). \tag{10}$$

Next we interchange the summations and break the permutation z down into $(z_1, \ldots, z_{i-1})$ corresponding to previous insertions, z_i corresponding to the next insertion, and $(z_{i+1}, \ldots, z_n)$ corresponding to future insertions, giving

$$\mathcal{L}(x) = \sum_{i=1}^{n} \sum_{z \in S_n} p(z) \log p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}) \tag{11}$$

$$= \sum_{i=1}^{n} \sum_{z_{1:i-1}} \sum_{z_i} \sum_{z_{i+1:n}} p(z) \log p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}) \tag{12}$$

$$= \sum_{i=1}^{n} \sum_{z_{1:i-1}} p(z_{1:i-1}) \sum_{z_i} p(z_i \mid z_{1:i-1}) \log p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}) \sum_{z_{i+1:n}} p(z_{i+1:n} \mid z_{1:i}) \tag{13}$$

$$= \sum_{i=1}^{n} \sum_{z_{1:i-1}} p(z_{1:i-1}) \sum_{z_i} p(z_i \mid z_{1:i-1}) \log p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1}), \tag{14}$$

where the simplification in the last line follows from the fact that $\sum_{z_{i+1:n}} p(z_{i+1:n} \mid z_{1:i}) = 1$.

From here, we can multiply and divide the outer sum by n to turn it into a mean, then arrive at the following simple sampling procedure to compute an unbiased estimate of our lower bound $\mathcal{L}(x)$ on the log-likelihood for a single example:

1. Sample a generation step $i \sim \text{Uniform}([1, n])$.
2. Sample a partial permutation $z_{1:i-1} \sim p(z_{1:i-1})$ for the first $i - 1$ insertions.
3. Compute a weighted sum over the next-step losses $\log p((c_i^z, l_i^z) \mid x_{1:i-1}^{z,i-1})$ scaled by the weighting distribution $p(z_i \mid z_{1:i-1})$ and the sequence length n.

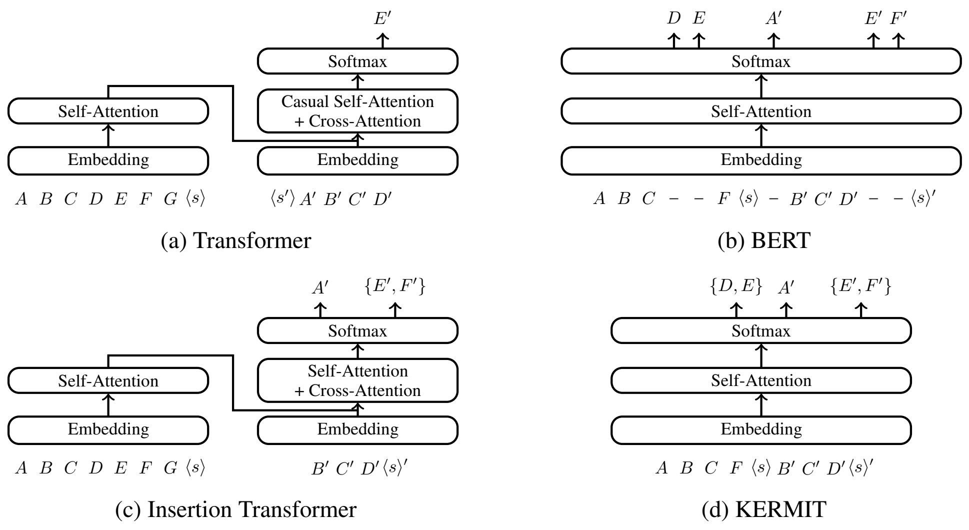

Figure 2: Diagram of various models. The Transformer (a) model predicts the next right token given the left context. The BERT (b) model predicts what is missing in the blank slots given the context. The Insertion Transformer (c) model predicts where and what is missing given the context. The KERMIT (d) model is an generalization of (c) where the context is over multiple sequences.

3.2 Inference

Using this model, inference can be autoregressive via greedy decoding

$$(\hat{c}, \hat{l}) = \operatorname*{argmax}_{c,l} p(c, l | \hat{x}_t) \tag{15}$$

or partially autoregressive via parallel decoding

$$\hat{c}_l = \operatorname*{argmax}_{c} p(c \mid l, \hat{x}_t). \tag{16}$$

In the case of parallel decoding, we perform simultaneous insertions at all non-finished slots. If we use a balanced binary tree prior for $p(z)$ (Stern et al., 2019), we can even achieve an empirical runtime of $\approx \log_2 n$ iterations to generate n tokens. One key advantage of insertion-based models over MLMs is that the output canvas can dynamically grow in size, meaning the length does not need to be chosen before the start of generation.

3.3 Pairs of Sequences

Thus far, we have discussed KERMIT for single sequences. We can easily extend KERMIT to pairs of sequences by directly modeling (x, y) as a concatenation of two sequences, $(x, y) = (x_1, \ldots, x_n, y_1, \ldots, y_m)$. For example, let our first sequence be $x = (A, B, C, \langle\text{EOS}\rangle)$ and our second sequence be $y = (A', B', C', D', E', \langle\text{EOS}\rangle)$. The concatenated sequence would then be $(x, y) = (A, B, C, \langle\text{EOS}\rangle, A', B', C', D', E', \langle\text{EOS}\rangle)$. With this approach, we can model pairs of sequences as if they were single sequences. Moreover, unlike seq2seq, our model is symmetric with regards to its treatment of the source and target, making it a strong candidate for extensions to multimodal data settings in future work.

By keeping our architecture order-agnostic and marginalizing over all possible orders in our training objective, KERMIT is able to learn the joint distribution and all its decompositions, including the marginals $p(x)$ and $p(y)$ and conditionals $p(y \mid x)$ and $p(x \mid y)$. We can also perform targeted training. More explicitly, if the model is provided with a canvas that fully contains x or y, then it will learn a conditional distribution. If the model is provided with an example where x or y is empty, then it will learn the opposing marginal distribution.

3.4 Model

We implement KERMIT as a single Transformer decoder stack (Vaswani et al., 2017), without any form of causal masking. The full self-attention mechanism allows the model to capture any rela-

Model	$\leftrightarrow$	En $\rightarrow$ De	De $\rightarrow$ En	Iterations
Autoregressive				
Transformer (Vaswani et al., 2017)	✗	27.3		n
Transformer (Our Implementation)	✗	27.8	31.2	n
Non-Autoregressive				
NAT (Gu et al., 2018)	✗	17.7	21.5	1
Iterative Refinement (Lee et al., 2018)	✗	21.6	25.5	10
Blockwise Parallel (Stern et al., 2018)	✗	27.4		$\approx n/5$
Insertion Transformer (Stern et al., 2019)	✗	27.4		$\approx \log_2 n \ll 10$
KERMIT				
Unidirectional ($p(y \mid x)$ or $p(x \mid y)$)	✗	27.8	30.7	$\approx \log_2 n \ll 10$
Bidirectional ($p(y \mid x)$ and $p(x \mid y)$)	✓	27.2	27.6	$\approx \log_2 n \ll 10$
Joint ($p(x, y)$)	✓	25.6	27.4	$\approx \log_2 n \ll 10$
+ Marginal Refining ($p(x)$ and $p(y)$)	✓	25.8	28.6	$\approx \log_2 n \ll 10$
$\hookrightarrow$ Unidirectional Finetuning	✗	28.7	31.4	$\approx \log_2 n \ll 10$
$\hookrightarrow$ Bidirectional Finetuning	✓	28.1	28.6	$\approx \log_2 n \ll 10$

Table 2: WMT English $\leftrightarrow$ German newstest2014 BLEU. Models capable of translating in both directions are marked with $\leftrightarrow$.

tionships between the input canvas and the predicted insertion operations with a constant number of operations. We follow Stern et al. (2019) and model the (content, location) distribution $p(c, l)$ as a factorized distribution $p(c, l) = p(c \mid l)p(l)$, where $p(c \mid l)$ is the standard Transformer softmax over the vocabulary, and a $p(l)$ is a softmax over the locations. Figure 2 visualizes the differences between a standard Transformer (Vaswani et al., 2017), BERT (Devlin et al., 2019), Insertion Transformer (Stern et al., 2019) and KERMIT.

4 Experiments

We perform experiments with KERMIT on the tasks of machine translation, self-supervised representation learning, and zero-shot cloze question answering.

4.1 Machine Translation

We first apply KERMIT on the competitive WMT 2014 English $\leftrightarrow$ German translation task. We follow the hyperparameter settings of the base Transformer (Vaswani et al., 2018). However, since KERMIT does not have an encoder, we simply double the decoder width. We perform no additional hyperparameter tuning. We also follow prior work (Gu et al., 2018; Stern et al., 2018, 2019; Lee et al., 2018) in using distillation (Hinton et al., 2015; Kim and Rush, 2016) to train our models. We follow Stern et al. (2019) in using a balanced binary tree loss, and we similarly observe an empirically logarithmic number of generation steps in sequence length when using parallel decoding. However, unlike Stern et al. (2019) we did not need to tune an EOS penalty, but simply set it to zero for all experiments.

We train several different KERMIT models for translation. First we train two unidirectional models, where the model observes a full source sentence (i.e., English or German) and is asked to generate the corresponding target sentence (i.e., German or English). These separately learn the conditional distributions $p(y \mid x)$ and $p(x \mid y)$, mimicking the traditional conditional generation setup. On the WMT 2014 test set, we achieve 27.8/30.7 BLEU with this approach, roughly matching our base Transformer baseline of 27.8/31.2 BLEU. We also train a bidirectional model on the union of the two unidirectional training sets, yielding a single model that captures both conditional distributions $p(y \mid x)$ and $p(x \mid y)$. We do not change any hyperparameters when training this model (i.e., we do not increase model capacity). The combined approach obtains 27.2/27.6 BLEU, nearly matching the baseline for English $\rightarrow$ German but falling slightly behind in the reverse direction.

We also train a full joint model that captures the full joint distribution $p(x, y)$ and factorizations thereof. Like the bidirectional model, the joint model can translate in either direction, but it can

Input: In order to develop such a concept, the town is reliant on the cooperation of its citizens.

Predicted: Um ein solches Konzept zu entwickeln, ist die Stadt auf die Zusammenarbeit ihrer Bürger angewiesen.

Parallel decode:

Um_ ein_ solches_ Konzept_ zu_ entwickeln_ , _ ist_ die_ Stadt_ auf_ die_ Zusammenarbeit_ ihrer_ Bürger_ angewiesen_ ._

Um_ ein_ solches_ Konzept_ zu_ entwickeln_ , _ ist_ die_ Stadt_ auf_ die_ Zusammenarbeit_ ihrer_ Bürger_ angewiesen_ ._

Um_ ein_ solches_ Konzept_ zu_ entwickeln_ , _ ist_ die_ Stadt_ auf_ die_ Zusammenarbeit_ ihrer_ Bürger_ angewiesen_ ._

Um_ ein_ solches_ Konzept_ zu_ entwickeln_ , _ ist_ die_ Stadt_ auf_ die_ Zusammenarbeit_ ihrer_ Bürger_ angewiesen_ ._

Um_ ein_ solches_ Konzept_ zu_ entwickeln_ , _ ist_ die_ Stadt_ auf_ die_ Zusammenarbeit_ ihrer_ Bürger_ angewiesen_ ._

Input: Frühere Gespräche zwischen den Parteien haben nur wenig zur Beilegung der Spannungen beigetragen, die durch eine Reihe von Zusammenstößen in diesem Jahr befeuert wurden.

Predicted: Previous talks between the parties have done little to resolve the tensions fueled by a series of clashes this year.

Parallel decode:

Prev ious_ talks_ between_ the_ parties_ have_ done_ little_ to_ resolve_ the_ tensions_ fueled_ by_ a_ series_ of_ cla she s_ this_ year_ ._

Prev ious_ talks_ between_ the_ parties_ have_ done_ little_ to_ resolve_ the_ tensions_ fueled_ by_ a_ series_ of_ cla she s_ this_ year_ ._

Prev ious_ talks_ between_ the_ parties_ have_ done_ little_ to_ resolve_ the_ tensions_ fueled_ by_ a_ series_ of_ cla she s_ this_ year_ ._

Prev ious_ talks_ between_ the_ parties_ have_ done_ little_ to_ resolve_ the_ tensions_ fueled_ by_ a_ series_ of_ cla she s_ this_ year_ ._

Prev ious_ talks_ between_ the_ parties_ have_ done_ little_ to_ resolve_ the_ tensions_ fueled_ by_ a_ series_ of_ cla she s_ this_ year_ ._

Figure 3: Example parallel decodes using KERMIT for English $\rightarrow$ German and German $\rightarrow$ English translation. In each row, the blue underlined tokens are those being inserted, and the gray tokens are those from the final output that have not yet been generated. Empirically, KERMIT requires only $\approx \log_2 n$ steps to generate a sequence of length n when trained with a balanced binary tree prior.

additionally be used for sampling or completing partial inputs. We use the same hyperparameter set as before. Since the model is now faced with a much more challenging task, it does slightly worse when limited to the same model size, but still reaches a respectable 25.6/27.4 BLEU. Unlike the previous models, however, we can incorporate monolingual data into the joint model's training setup to supplement its knowledge of the marginals $p(x)$ and $p(y)$. We accordingly train a joint model with all our paired data and 1M additional samples of English and German monolingual data randomly selected from the WMT 2014 monolingual corpus. Without altering model capacity, we find that refining the marginals gives us a 1.2 BLEU improvement on German $\rightarrow$ English. Finally, we take the model which was trained on the full joint distribution with marginal refinement, and further finetune it on both the unidirectional and bidirectional settings. We find a small improvement in BLEU over the original models in both settings.

Table 2 summarizes our results. We emphasize that virtually all of our models outperform prior non-fully-autoregressive approaches in terms of BLEU. We also note that the observed number of iterations required to generate n tokens is roughly $\log_2 n$ due to the use of a balanced binary tree loss and parallel decoding, which is substantially lower than autoregressive models which require n steps. Some examples of parallel decodes are shown in Figure 3. Our models require an average of 5.5-6.5 decoding iterations for the sentences in the test set, outperforming the constant-time models of Lee et al. (2018) which require 10 iterations in both BLEU and empirical decoding complexity.

We also draw samples from the model to highlight its infilling and generation capabilities. Figure 4 captures some examples. We first show unconditional sampling of an (English, German) sentence pair. We also take a translation example from the newstest2013 dev set and split it in half, sampling completions after seeding the English side with the first half and the German side with the second half. We find the model is capable of generating a very diverse set of coherent samples.

4.2 Representation Learning

Like its close friend BERT (Devlin et al., 2019), KERMIT can also be used for self-supervised representation learning and applied to various language understanding tasks. We follow the same training procedure and hyperparameter setup as BERT$_{\text{LARGE}}$. However, instead of masking 15% of the tokens and replacing them with blank tokens like in BERT (Devlin et al., 2019), KERMIT simply drops them out completely from the sequence.

No seeding (unconditional):

English: Nonetheless, we feel, with fury, at the fact that the 500 million have no contradiction on a common approach to productivity.
German: Dennoch sind wir mit Wut der Ansicht, dass die 500 Millionen keinen Widerspruch in einem gemeinsamen Produktivitätsansatz aufweisen.

. . . .

English Groundtruth: - Please tell us, in simple terms, about the work your research group does.
German Groundtruth: - Beschreiben Sie bitte kurz, welche Forschungen Ihre Gruppe betreibt.

English Seed: - Please tell us, in simple terms
German Seed: welche Forschungen Ihre Gruppe betreibt.

English: - Please tell us what sort of research your group is conducting, in simple terms.
German: - Bitte teilen Sie uns einfach mit, welche Forschungen Ihre Gruppe betreibt.

English: - Please tell us, in quite simple terms, what kind of research your group operates.
German: - Bitte teilen Sie uns in ganz einfach mit, welche Forschungen Ihre Gruppe betreibt.

English: - Please tell us what research, in simple terms, what your group actually runs.
German: - Bitte sagen Sie uns ganz einfach, welche Forschungen Ihre Gruppe eigentlich betreibt.

English: - Please, tell us what research your group is doing, in more simple terms.
German: - Bitte sagen Sie uns, welche Forschungen Ihre Gruppe einfacher betreibt.

English: - Please tell us what your group will be conducting public research on, in simple terms.
German: - Bitte teilen Sie uns einfach mit, welche Forschungen Ihre Gruppe betreibt.

English: - Please tell us, what sort of research your group is undertaking in simple terms.
German: - Bitte sagen Sie uns, welche Forschungen Ihre Gruppe betreibt.

. . . .

Figure 4: Paired translation samples drawn from KERMIT, with and without seeding the initial canvas with text. In the bottom portion of the figure, the seed text is shown in gray, and different continuations sampled from the model are shown in black. We emphasize the diversity of generation.

Model	**Generative?**	**CoLA**	**SST-2**	**MRPC**	**STS-B**	**QQP**	**MNLI-(m/mm)**	**QNLI**	**RTE**	**WNLI**	**AX**	**Score**
GPT (Radford et al., 2018)	✓	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	87.4	56.0	53.4	29.8	72.8
BERT (Devlin et al., 2019)	✗	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
KERMIT	✓	60.0	94.2	88.6/84.3	86.6/85.6	71.7/89.0	85.6/85.2	92.0	68.4	65.1	37.6	79.8

Table 3: GLUE benchmark scores (as computed by the GLUE evaluation server). Of these models, only GPT and KERMIT admit a straightforward generation process.

Prior to BERT, the best representation learning approach was to use a language model such as GPT (Radford et al., 2018). BERT outperforms GPT in large part because of its *deeply bi-directional* architecture, but in the process BERT sacrifices the ability to perform straightforward generation. While we find KERMIT to perform slightly behind BERT, KERMIT maintains the ability to generate text while obtaining results that are much closer to BERT rather than GPT. The GLUE benchmark (Wang et al., 2019) results are summarized in Table 3.

4.3 Zero-Shot Cloze Question Answering

Finally, we also investigate the infilling abilities of KERMIT and related approaches by evaluating their performance on zero-shot cloze question answering. In particular, we aim to understand how effective these models are for fill-in-the-blank-style question answering after being trained only on language modeling data without any task-specific fine-tuning.

For this experiment, we use the human-annotated QA2D dataset assembled by Demszky et al. (2018), which consists of examples from the SQuAD dataset (Rajpurkar et al., 2016) in which the answer has been extended from a single phrase into a full declarative sentence. These can be transformed into cloze instances by removing the answer phrase from the declarative output. For example, given the question "When was Madonna born?" and the answer "August 16, 1958", the full declarative answer would be "Madonna was born on August 16, 1958.", and the associated cloze instance would be "Madonna was born on ______ ."

Plymouth has a post-war shopping area in the city centre with substantial pedestrianisation. At the west end of the zone inside a grade II listed building is the Pannier Market that was completed in 1959 – pannier meaning "basket" from French, so it translates as "basket market". In terms of retail floorspace, Plymouth is ranked in the top five in the South West, and 29th nationally. Plymouth was one of the first ten British cities to trial the new Business Improvement District initiative. The Tinside Pool is situated at the foot of the Hoe and became a grade II listed building in 1998 before being restored to its 1930s look for £3.4 million. **What notable location was named a grade II listed building in 1998? ___ was named a grade II listed building in 1998 .**

Model	**Answer**
GPT-2	→ "Plymouth"
+ Oracle Length	→ "A listed building"
BERT	→ "plymouth"
+ Oracle Length	→ ": the pool"
KERMIT	→ **"the tinside pool"**
Correct	→ "Tinside Pool"

Figure 5: Example of KERMIT, BERT and GPT-2 performing zero-shot cloze question answering on SQuAD. The question and cloze question are bolded. Note that BERT and GPT-2 prefer a shorter, incorrect answer, unless given the oracle answer length.

We take the KERMIT model trained from Section 4.2 and two powerful language models (BERT (Devlin et al., 2019) and the largest public version of GPT-2[2] (Radford et al., 2019)), and evaluate their ability to fill in the blank of each cloze instance, each without specifically being trained on data of this form. We employ different decoding strategies as required for each model, detailed below.

KERMIT We split the passage in half and present KERMIT with examples of the form

```
[CLS] passage(1/2) [SEP] passage(2/2) question cloze(left) cloze(right) [SEP]
```

where `cloze(left)` and `cloze(right)` are the portions of the declarative answer before and after the gap. Since KERMIT can natively perform insertions, we simply perform a parallel decode constrained to take place within the gap and extract the output as our answer.

BERT We split the passage in half and present BERT with examples of the form

```
[CLS] passage(1/2) [SEP] passage(2/2) question cloze(left) [MASK]*n cloze(right) [SEP]
```

Here we include explicit `[MASK]` tokens, running separate decodes with $n = 1, 2, \ldots$ up to 4 or the oracle answer length, whichever is greater. We then choose the one with the highest score under the model and extract the outputs at the masked positions as the answer. Each decode consists of a beam search in which one `[MASK]` is filled at a time. For each element on the beam, we choose the remaining `[MASK]` position with the highest confidence (lowest entropy) as the next position to fill. We found that this beam-search did substantially better than left-to-right decoding or parallel decoding.

GPT-2 For GPT-2, a left-to-right language model, we cannot directly condition on both the left and right context. Instead, we first present the model with the prefix

```
passage question cloze(left)
```

and sample continuations of varying lengths. For each continuation, we then append `cloze(right)` and compute the score of the full sequence under the model. We select the best-scoring sequence and extract the portion in the gap as the answer. To efficiently obtain continuations of varying lengths, we generate 20 extended continuations from the model, then treat all prefixes of those continuations as candidate values to go in the gap.

We evaluate on 50,000 cloze-formulated questions from SQuAD, using the standard SQuAD evaluation script to compute accuracy in terms of exact match and token-level F1. Results are presented in Table 4. KERMIT performs significantly better on this zero-shot cloze task than the other two approaches thanks to its infilling capabilities learned through its insertion-oriented objective, achieving 30.3 F1 and 20.9% exact match. BERT's performance falls short of KERMIT, as it often prefers shorter completions since it is not required to handle length modeling

Model	**Exact Match**	**F1**
GPT-2	10.9	16.6
+ Oracle Length	12.2	18.2
BERT	12.3	18.9
+ Oracle Length	16.2	23.1
KERMIT	**20.9**	**30.3**

Table 4: SQuAD zero-shot cloze question answering.

[2]The 345M parameter "medium size" model.

during training. GPT-2 lags further behind the others due to its inability to condition on the context on both sides of the gap during inference. Even when the oracle length (i.e., the ground-truth length of the answer) is provided to BERT and GPT-2, KERMIT still substantially outperforms all other models.

5 Conclusion

In this paper, we present KERMIT, an insertion-based framework for sequences that can model the joint data distribution and its decompositions (i.e., marginals and conditionals). KERMIT can generate text in an arbitrary order – including bidirectional machine translation and cloze-style infilling – and empirically can generate sequences in logarithmic time. It uses a simple neural architecture that can additionally produce contextualized vector representations of words and sentences. We find KERMIT is capable of matching or exceeding state-of-the-art performance on three diverse tasks: machine translation, representation learning, and zero shot cloze question answering.

Acknowledgments

We give thanks to Samy Bengio, Zhifeng Chen, Jamie Kiros, Luheng He, Geoffrey Hinton, Quoc Le, Lala Li, Mohammad Norouzi, Yu Zhang, and the Google Brain team for useful discussions and technical assistance. Special thanks to Jamie Kiros for brainstorming the name KERMIT.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-End Attention-based Large Vocabulary Speech Recognition. In *ICASSP*.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition. In *ICASSP*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Demszky, D., Guu, K., and Liang, P. (2018). Transforming Question Answering Datasets Into Natural Language Inference Datasets. In *arXiv*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. (2018). Non-Autoregressive Neural Machine Translation. In *ICLR*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

Kim, Y. and Rush, A. M. (2016). Sequence-Level Knowledge Distillation. In *EMNLP*.

Lee, J., Mansimov, E., and Cho, K. (2018). Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. In *EMNLP*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. In *arXiv*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. (2019). Insertion Transformer: Flexible Sequence Generation via Insertion Operations. In *ICML*.

Stern, M., Shazeer, N., and Uszkoreit, J. (2018). Blockwise Parallel Decoding for Deep Autoregressive Models. In *NeurIPS*.

Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H., and Wu, H. (2019). ERNIE: Enhanced Representation through Knowledge Integration. In *arXiv*.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to Sequence Learning with Neural Networks. In *NIPS*.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, L., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N., and Uszkoreit, J. (2018). Tensor2Tensor for Neural Machine Translation. In *AMTA*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. In *NIPS*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and Tell: A Neural Image Caption Generator. In *CVPR*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A. (2017). Tacotron: Towards End-to-End Speech Synthesis. In *INTERSPEECH*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*.