MASKED TRANSLATION MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce the masked translation model (MTM) which combines encoding and decoding of sequences within the same model component. The MTM is based on the idea of masked language modeling and supports both autoregressive and non-autoregressive decoding strategies by simply changing the order of masking. In experiments on the WMT 2016 Romanian→English task, the MTM shows strong constant-time translation performance, beating all related approaches with comparable complexity. We also extensively compare various decoding strategies supported by the MTM, as well as several length modeling techniques and training settings.

1 INTRODUCTION

Neural machine translation (NMT) has been developed under the encoder-decoder framework (Sutskever et al., 2014) with an intermediary attention mechanism (Bahdanau et al., 2015). The encoder learns contextualized representations of source tokens, which are used by the decoder to predict target tokens. These two components have individual roles in the translation process, and they are connected via an encoder-decoder attention layer (Bahdanau et al., 2015). Many advances in NMT modeling are based on changes in the internal layer structure (Gehring et al., 2017; Wang et al., 2017; Vaswani et al., 2017; Chen et al., 2018; Dehghani et al., 2019; Yang et al., 2019), tweaking the connection between the layers (Zhou et al., 2016; Shen et al., 2018; Bahar et al., 2018; Li et al., 2019a), or appending extra components or latent variables (Gu et al., 2016; Zhang et al., 2016; Shah & Barber, 2018; Hao et al., 2019) – all increasing the overall architectural complexity of the model while keeping the encoder and decoder separated.

Our goal is to simplify the general architecture of machine translation models. For this purpose, we propose the masked translation model (MTM) – a unified model which fulfills the role of both the encoder and decoder within a single component. The MTM gets rid of the conventional decoder as well as the encoder-decoder attention mechanism. Its architecture is only a sequence encoder with self-attention layers, trained with an objective function similar to masked language modeling (Devlin et al., 2019). In order to model the translation problem, the MTM is given the concatenation of the source and target side from a parallel sentence pair. This approach is similar to the translation language model presented by Lample & Conneau (2019), but focuses on the target side, i.e. the masking is applied to some selected positions in the target sentence. The MTM is trained to predict the masked target words relying on self-attention layers which consider both the source sentence and a masked version of the target sentence. Trained in this way, the model is perfectly suitable for non-autoregressive decoding since the model learned to predict every position in parallel, removing the dependency on decisions at preceding target positions.

Within its extremely simple architecture, one can realize various decoding strategies, e.g., using left-to-right, non-autoregressive, or iterative decoding by merely adjusting the masking schemes in search. We present a unified formulation of the MTM for different decoding concepts by factorizing the model probability over a set of masked positions.

The MTM has several advantages over the conventional encoder-decoder framework:

- A simpler architecture
- Source and target representations interact in every layer
- A variety of decoding strategies including constant-time approaches (Section 3.3.1)

On the WMT 2016 Romanian→English translation task, our MTM achieves a better performance than comparable non-autoregressive/constant-time methods while keeping its simple architecture. Using our general formulation of the MTM, we compare the translation performance of various decoding strategies. Moreover, we show that this model allows for decoding speed-up by merely adjusting the number of iterations at the small cost of translation performance.

2 Related Work

There have been some attempts to combine the encoder and decoder into a single component for simplified translation modeling. He et al. (2018) share the encoder and decoder parameters of a Transformer translation model (Vaswani et al., 2017) and allow the encoder-decoder attention to access the inner layers of the encoder as well. Fonollosa et al. (2019) extend this idea by adding locality constraints in all attention layers. Radford et al. (2019) train a Transformer decoder on a large monolingual corpus as a language model and use it as an unsupervised translation model on pairs of source and target sentences. Similarly to our work, all these approaches couple the encoding and decoding on the self-attention level. However, their decoding considers only left-side target context, enabling only left-to-right autoregressive translation. Furthermore, their encoding of a source sentence is limited to the source side itself, while our MTM can refine the source representations according to a partial target hypothesis represented in the decoder states. Both aspects hinder their methods from making the best use of the bidirectional representation power of the combined model.

Non-autoregressive NMT, which predicts all target words in parallel, potentially exploits full bidirectional context in decoding. To make the parallel decoding produce a reasonable hypothesis, Gu et al. (2018) reuse the source words as inputs to the decoder and insert an additional attention module on the positional embeddings. Lee et al. (2018) use a separate decoder to revise the target hypotheses iteratively, where Ghazvininejad et al. (2019) train a single decoder with MLM objectives for both the first prediction and its refinements. To improve the integrity of the hypotheses, one could also employ an autoregressive teacher to guide the states of the non-autoregressive decoder (Wei et al., 2019; Li et al., 2019b), apply sentence-level rewards in training (Shao et al., 2019), or integrate generative flow latent variables (Ma et al., 2019). The self-attention layers of their decoders attend to all target positions, including past and future contexts. However, all these methods still rely on the encoder-decoder framework. In this work, we collapse the boundary of the encoding and decoding of sequences and realize non-autoregressive NMT with a unified model. Regardless of the encoding or decoding, the self-attention layers of our MTM attend to all available source and target words for flexible information flow and a model of simplicity.

A common problem in non-autoregressive sequence generation is that the length of an output should be predefined beforehand. The problem has been addressed by averaging length difference (Li et al., 2019b), estimating fertility (Gu et al., 2018), dynamically inserting blank outputs via connectionist temporal classification (CTC) (Libovický & Helcl, 2018), or directly predicting the total length from the encoder representations (Lee et al., 2018; Ghazvininejad et al., 2019; Wei et al., 2019; Ma et al., 2019). In this work, we train a separate, compact length model on given bilingual data.

The training of an MTM is based on the MLM objective (Devlin et al., 2019), developed for pre-training representations for natural language understanding tasks (Wang et al., 2019). Lample & Conneau (2019) concatenate source and target sentences and use them together as the input to an MLM, where both source and target tokens are randomly masked for the training. This improves cross-lingual natural language inference in combination with the original MLM objective, but has not been applied to translation tasks. We use the concatenated input sentences but selectively mask out only target tokens to implement source→target translation. As for inference with an MLM, Wang & Cho (2019) and Ghazvininejad et al. (2019) propose to build up an output sequence iteratively by adjusting the input masking for each iteration. In this work, the generation procedures of both works are tested and compared within our MTM, along with other autoregressive/non-autoregressive decoding strategies (Section 4.1).

3 Masked Translation Model

We introduce the masked translation model (MTM) focusing on three aspects: model architecture, training, and decoding. In the corresponding sections, we show how the MTM 1) relaxes the con-

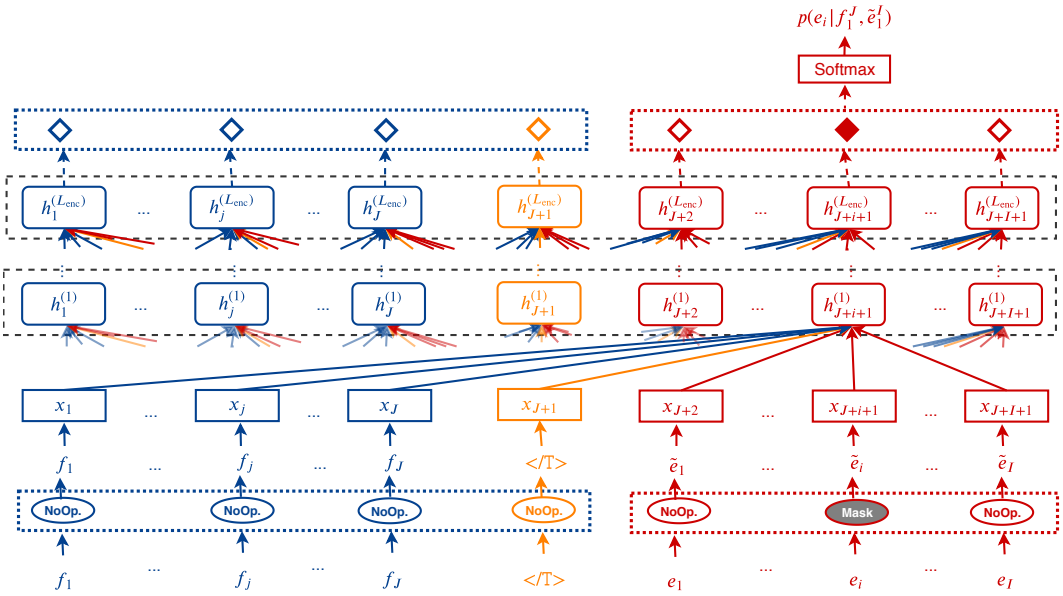

Figure 1: Visualization of an MTM model during training. The architecture is equivalent to a standard Transformer encoder with bidirectional self-attention. Additionally, the corruption decisions for each position are illustrated in an oval above the model input. The corrupted positions M are indicated before the softmax layer by a filled diamond. Non-corrupted positions (blank diamond) are not predicted in training.

ventional architectural constraint of the encoder-decoder framework, 2) learns to perform both non-autoregressive translation and its refinements, and 3) does translation with various decoding strategies in a variable number of iterations.

3.1 Model

Given a source sentence $f_1^J = f_1, ..., f_j, ..., f_J$ (source input) the goal of an MTM p_θ is to generate the correct translation $e_1^I = e_1, ..., e_i, ..., e_I$ (target output) by modeling:

$$p(e_1^I|f_1^J) = \prod_{i=1}^{I} p(e_i|f_1^J, e_1^{i-1}, e_{i+1}^I) = \prod_{i=1}^{I} p_\theta(e_i|f_1^J, \tilde{E}_i) \tag{1}$$

where $\tilde{E}_i := \tilde{E}(e_1^{i-1}, e_{i+1}^I)$ (target input) is a corrupted version – a subset – of the surrounding context in the target sentence e_1^I. Therefore, the MTM models the true target sentence e_1^I independently for each position, given both the true source and a noisy version of the target context.

Figure 1 illustrates the MTM network architecture which is the same as the MLM presented by Devlin et al. (2019) or the encoder of a Transformer network (Vaswani et al., 2017) with an output softmax layer on top of all corrupted positions. In particular, an MLM consists of N transformer encoder layers each containing two blocks: a self-attention layer with full bidirectional context as well as two linear layers with a ReLU activation in between. Layer normalization is applied before every block, and a residual connection is used to combine the output of a block with its input (Vaswani et al., 2017).

The source f_1^J and the noisy target sentence $\tilde{e}_1^I$ are concatenated with a special boundary token '</T>' as separator and given to the model as a single input sequence. The tokens of this sequence are converted to embedding vectors x_1^{J+I+1}, whose space is shared over the source and target languages by a joint subword vocabulary. A positional encoding vector (Vaswani et al., 2017) is added where the position index is reset at the first target token. Similarly to He et al. (2018) and Lample & Conneau (2019), we add language embeddings to distinguish source and target vocabularies efficiently. The embedded representations pass through N transformer encoder layers, where the attention has no direction constraints; this allows the model to use the full bidirectional context beyond

the language boundary. Note that the hidden representations of source words can also attend to those of (partially hypothesized) target tokens, which is impossible in the encoder-decoder architecture.

3.2 TRAINING

During training the target side input $\tilde{e}_1^I$ may be 1) **fully masked out**, resembling the initial stage of translation before hypothesizing any target words, 2) **partially corrupted**, simulating intermediate hypotheses of the translation process that need to be corrected, 3) or even the **original target sentence** e_1^I, representing the final stage of translation which should not be refined further. The different levels of corruptions are used to model all plausible cases which we encounter in the decoding process – from primitive hypotheses to high-quality translations (Section 3.3).

Given bilingual training data $\mathcal{D} = \{(f_1^J, e_1^I)\}$, the first scenario can be easily simulated by masking out all positions of the target input, i.e. $\tilde{e}_i = \texttt{</M>}$ for all i, while the target output e_1^I remains unchanged. In this case, the model effectively describes the conditional distribution $p(e_1^I | f_1^J, I)$. Note that a model for length prediction is trained separately and the details of which can be found Appendix A.

We cannot expect a single step of parallel decoding to output an optimal translation from only the source context. Therefore, we consider the second scenario of refining the hypothesis, where we simulate a partial hypothesis in training by artificially corrupting the given target sentence in the input. For training an MTM, we formulate this corruption by probabilistic models to 1) select the positions to be corrupted (p_s) and 2) decide how such a position is corrupted (p_c). The goal of the MTM is now to reconstruct the original target sentence e_1^I. This leads to the training loss:

$$\mathcal{L}(\theta; \mathcal{D}) = - \mathop{\mathbb{E}}_{(f_1^J, e_1^I) \in \mathcal{D}} \left[\mathop{\mathbb{E}}_{C \sim p_\text{s}(C|e_1^I)} \left[\mathop{\mathbb{E}}_{\substack{\tilde{e}_1^I : \tilde{e}_i \sim p_\text{c}(\tilde{e}_i|e_i) \\ \forall i \in C}} \left[\frac{1}{|C|} \sum_{i \in C} \log p_\theta(e_i | f_1^J, \tilde{e}_1^I) \right] \right] \right] \tag{2}$$

where C is a set of positions to be corrupted. The corrupted target input $\tilde{e}_1^I$ is generated in two steps of random decisions (Devlin et al., 2019):

1. Target positions $i \in C$ for the corruption are sampled from a uniform distribution until $\lceil \rho_\text{s} \cdot I \rceil$ samples are drawn, with hyperparameter $\rho_\text{s} \in [0, 1]$. We denote this selection process by $C \sim p_\text{s}(C|e_1^I) = p_\text{s}(C|I)$ as a simplified notation.
2. The specific type of corruption for each selected position $i \in C$ is chosen independently by sampling $\tilde{e}_i$ from $p_\text{c}(\tilde{e}_i|e_i)$.

Note that we train the model to reconstruct the original token e_i only for the corrupted positions $i \in C$. For the remaining positions $i' \notin C$, the corrupted sentence is filled with the original word, i.e. $\tilde{e}_{i'} = e_{i'}$. These uncorrupted positions provide the context to the network for the denoising, and no loss is applied for these positions to prevent a bias towards copying. We optimize this criterion in a stochastic way, where C and the $\tilde{e}_i$ are sampled anew for each epoch and each training instance. In principle, the MTM is a denoising autoencoder of the target sentence conditioned on a source sentence.

The MTM training can be customized by selecting p_s and p_c appropriately. Following Devlin et al. (2019), the probability p_s is defined as a uniform distribution over all target positions, without considering the content of the sentence. For the corruption model p_c, we define a set of operations and assign a probability mass $\rho_o \in [0, 1]$ to each operation o. We use the three operations presented by Devlin et al. (2019):

- Replace with a mask token:

$$\textit{Mask:} \qquad p_\text{c}(\tilde{e}_i = \texttt{</M>} | e_i) = \rho_\text{mask} \tag{3}$$

- Replace with a random word e_* uniformly sampled from the target vocabulary V_e:

$$\textit{Random:} \qquad p_\text{c}(\tilde{e}_i = e_* \sim V_e | e_i) = \rho_\text{rand} \tag{4}$$

- Keep unchanged:

$$\textit{Keep:} \qquad p_\text{c}(\tilde{e}_i = e_i | e_i) = \rho_\text{keep} \tag{5}$$

Original: $e_1^I =$	Thanks	for	reading	this
Operation:	*NoOp.*	*Keep*	*Mask*	*Random*
Corrupted: $\tilde{e}_1^I =$	Thanks	for	</M>	cat
Loss:	×	✓	✓	✓

Figure 2: Example of the MTM corruption process on the target input sentence "Thanks for reading this" with resulting loss $\sum_{i=2}^{4} \log p_\theta(e_i | f_1^J, \tilde{e}_1^4)$.

Figure 2 shows an example of corrupting a target input sentence in the MTM training. Here all positions except 1 are corrupted, i.e. $C = \{2, 3, 4\}$. At Position 2 the original word is kept by the *Keep* operation of p_c but in contrast to Position 1 (*No Operation*) there is a training loss added for Position 2.

3.3 DECODING

As described above, the MTM is designed and trained to deal with intermediate hypotheses of varying quality as target input. Accordingly, decoding with an MTM consists of multiple iterations $\tau = 1, ..., T$: A non-autoregressive generation of the initial hypothesis (for $\tau = 1$) and several steps of iterative refinements (inspired by Lee et al. (2018)) of the hypothesis (for $\tau > 1$). In the context of this work, an iteration of the MTM during decoding refers to one forward pass of the model based on a given source and target input, followed by the selection of a target output based on the predictions of the MTM. To simplify the notation, we denote the given source sentence by $F = f_1^J$ and the generated target translation by $\hat{E} = \hat{e}_1^I$.

Similar to traditional translation models, the goal of decoding with an MTM is to find a hypothesis $\hat{E}$ that maximizes the translation probability:

$$F \mapsto \hat{E}(F) = \arg\max_{E,I} \{p(E, I|F)\} = \arg\max_{E,I} \{p(E|I, F) \cdot p(I|F)\} \tag{6}$$

We approximate this by a two stage maximization process where we first determine the most likely target length $\hat{I} := \arg\max_I \{p(I|F)\}$ followed by:

$$\hat{E}(F, \hat{I}) = \arg\max_{E} \{p(E|\hat{I}, F)\} \tag{7}$$

Instead of a left-to-right factorization of the target sentence which is common in autoregressive decoding, we perform a step-wise optimization on the whole sequence. For this we define the sequence $\hat{E}^{(\tau)}$ starting from $\hat{E}^{(0)} :=$ </M>, . . . , </M> by selecting the best hypothesis given the predecessor $\hat{E}^{(\tau-1)}$, i.e.:

$$\hat{E}^{(\tau)} := \arg\max_{E^{(\tau)}} p(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}). \tag{8}$$

Introducing an intermediate representation $\tilde{E}$ allows for a more fine-grained control of the decoding process:

$$p(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) := \sum_{\tilde{E}^{(\tau-1)}} p_\theta(E^{(\tau)}|\tilde{E}^{(\tau-1)}, F, \hat{I}) \cdot p_{\text{m}}(\tilde{E}^{(\tau-1)}|\hat{E}^{(\tau-1)}, F, \hat{I}) \tag{9}$$

where the probability p_θ is modeled by a neural network with parameters θ and the masked sequence $\tilde{E}^{(\tau-1)}$ is modelled by p_{m}, which defines a specific search strategy (Section 3.3.1). In most scenarios, the search strategy is defined to be deterministic, which has the effect that all probability mass of p_m is concentrated on one masked sequence $\tilde{E}^{(\tau-1)}$ and we can reformulate Equation (8) as:

$$\hat{E}^{(\tau)} = \arg\max_{E^{(\tau)}} p_\theta(E^{(\tau)}|\tilde{E}^{(\tau-1)}, F, \hat{I}). \tag{10}$$

and thus the score is defined solely by the MTM network.

The iterative procedure presented in Equation (10) describes a greedy optimization, as it selects the currently best hypothesis in each iteration. This does not provide any guarantee for the quality of

Algorithm 1: MTM Decoding

Input: Source sentence $F = f_1^J$
$\hat{I} \leftarrow \arg\max_I \{p(I|F)\}$
$\hat{E}^{(0)} \leftarrow$ </M>, ..., </M>
for $\tau \leftarrow 1, \ldots, T$ **do**
  Sample a masked sequence: $\tilde{E}^{(\tau-1)} \sim p_{\text{m}}(\tilde{E}^{(\tau-1)}|\hat{E}^{(\tau-1)})$
  Compute the model output greedily for every position:
    $\hat{e}_i^{(\tau)} \leftarrow \arg\max_{e_i} p_\theta(e_i|\tilde{E}^{(\tau-1)}, F, \hat{I}) \quad \forall i \in \{1, ..., \hat{I}\}$
  $\hat{E}^{(\tau)} \leftarrow \hat{e}_1^{(\tau)}, ..., \hat{e}_{\hat{I}}^{(\tau)}$
Output: $\hat{E} \leftarrow \hat{E}^{(T)}$

the final output. However, if we assume that the model score improves in each iteration, i.e. if we can show that:

$$\max_{E^{(\tau+1)}} p(E^{(\tau+1)}|\tilde{E}^{(\tau)}, F, \hat{I}) \geq \max_{E^{(\tau)}} p(E^{(\tau)}|\tilde{E}^{(\tau-1)}, F, \hat{I}) \;\; \forall \tau = 1, \ldots, T-1 \tag{11}$$

then we know that the maximum score is obtained in the last iteration T. To the best of our knowledge, it is not possible to provide a theoretical proof for this property, yet we will show empirical evidence that it holds in practice (see Section 4.1).

Thus, in order to find the best hypothesis, it is sufficient to follow the recursive definition presented in Equation (10), which can be computed straight forward resulting in an iterative decoding scheme. Algorithm 1 describes this process of MTM decoding. In short, 1) generate a hypothesis ($\hat{E}$), 2) select positions to be masked, and 3) feed the masked hypothesis ($\tilde{E}$) back to the next iteration.

Note that the output for each position $e_i^{(\tau)}$ is computed in the same forward pass without a dependency on other words $e_{i'}^{(\tau)}$ ($i' \neq i$) from the same decoding step. This means that the first iteration ($\tau = 1$) is non-autoregressive decoding (Gu et al., 2018; Libovický & Helcl, 2018). Non-autoregressive models tend to suffer from the multimodality problem (Gu et al., 2018), where conditionally independent models are inadequate to capture the highly multimodal distribution of target translations.

Our MTM decoding prevents this problem making iterative predictions $\hat{E}^{(\tau)}$ each conditioned on the previous sequence. Each $\hat{E}^{(\tau)}$ results from one forward pass, yielding a complexity of $\mathcal{O}(T)$ decoding steps instead of $\mathcal{O}(I)$ as in traditional autoregressive NMT. Thus the MTM decoding can potentially be much faster than conditional decoding of a standard encoder-decoder NMT model, as the number of iterations is not dictated by the target output length I. Furthermore, compared to the pure non-autoregressive decoding with only one iteration, our decoding algorithm may collapse the multimodal distribution of the target translation by conditioning on the previous output (Lee et al., 2018).

3.3.1 Decoding Strategies

The masking probability p_{m} introduced in Equation (9) resembles the two-step corruption of the training (Equation (2)):

$$p_{\text{m}}(\tilde{E}^{(\tau)}|\hat{E}^{(\tau)}) = p_{\text{s}}(C^{(\tau)}|C^{(\tau-1)}) \prod_{i \in C^{(\tau)}} p_{\text{c}}(\tilde{e}_i^{(\tau)}|\hat{e}_i^{(\tau)}) \tag{12}$$

where C is a set of positions to be masked. Similarly to the training, the corruption is performed only for $i \in C^{(\tau)}$ and the remaining positions $i' \notin C^{(\tau)}$ are kept untouched. For the corruption model p_c in decoding, only the *Mask* operation is activated, i.e. $\rho_{\text{mask}} = 1$ and $\rho_o = 0$ for $o \neq$ mask. This leads to the following simple decisions:

$$\tilde{e}_i^{(\tau)} = \begin{cases} \text{</M>}, & \text{if } i \in C^{(\tau)} \\ \hat{e}_i^{(\tau)}, & \text{otherwise} \end{cases} \tag{13}$$

The resulting masked sequence $\tilde{E}^{(\tau)}$ is supposed to shift the model's focus towards a selected number of words, chosen by the decoding strategy p_s.

Given this definition of p_{c} above, a masked (intermediate) hypothesis in decoding is determined solely by the position selection p_{s}, which differs by decoding strategy. Each decoding strategy starts from a fully masked target input, i.e. $C^{(0)} = \{1, ..., I\}$, and uncovers positions incrementally in each iteration.

Fully unmasking The simplest solution is to simply feed back the completely unmasked sequence in each iteration (Lee et al., 2018):

$$p_{\mathrm{s}}(C^{(\tau)}|C^{(\tau-1)}) = p_{\mathrm{s}}(C^{(\tau)}) = \begin{cases} 1, & \text{if } C^{(\tau)} = \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

This method works with the richest context from the output of the previous iteration. This may, however, hurt the model's performance as the first output is often quite poor, and the focus of the model is spread across the whole sentence.

Random Instead of unmasking all positions from the beginning, one can unmask the sequence randomly one position at a time (Wang & Cho, 2019), inspired by Gibbs sampling (Geman & Geman, 1984):

$$p_{\mathrm{s}}(C^{(\tau)}|C^{(\tau-1)}) = \begin{cases} \frac{1}{I}, & \text{if } C^{(\tau)} = C^{(\tau-1)} \backslash \{i\} \text{ with } i \in \{1, \ldots, I\} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Note that this method is nondeterministic and it takes at least I iterations before the output is conditioned on the completely unmasked sequence.

Left-to-Right (L2R) A deterministic alternative to the random strategy is to unveil the sequence step-wise, starting from the left-most position in the target sequence. In every decoding iteration $\tau = 1, ..., T$, the index $i = \tau - 1$ is removed from the set of masked positions:

$$p_{\mathrm{s}}(C^{(\tau)}|C^{(\tau-1)}) = \begin{cases} 1, & \text{if } C^{(\tau)} = C^{(\tau-1)} \backslash \{\tau - 1\} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

This corresponds to the traditional autoregressive NMT, but the parallel nature of our MTM decoding inherently enables to update the prediction for any position at any time, e.g. the prediction for the first position can change in the last iteration. Furthermore, it allows for different step-wise strategies – revealing the sequence **right-to-left (R2L)** or starting from the middle in both directions (**middle-out**) – without re-training the model.

Confidence-based L2R decoding ignores a huge advantage of the MTM, which is the property that the fully bidirectional model can predict sequence elements in any given order. This characteristic is leveraged by masking a decreasing number of $K(\tau)$ positions in each iteration:

$$p_{\mathrm{s}}(C^{(\tau)}|C^{(\tau-1)}) = \begin{cases} 1, & \text{if } C^{(\tau)} = \{i_1, ..., i_{K(\tau)}\} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

At each iteration, $K(\tau)$ positions with the lowest model score p_θ (confidence) remain masked:

$$i_k = \underset{i \notin \{i_1, ..., i_{k-1}\}}{\arg\min} \left\{ p_\theta(\hat{e}_i^{(\tau)} | \tilde{E}^{(\tau-1)}, F) \right\} \qquad \forall k \in \{1, ..., K(\tau)\} \tag{18}$$

where the number of masked positions $K(\tau)$ is chosen to be linearly decreasing over the number of iterations T (Ghazvininejad et al., 2019):

$$K(\tau) = I - \tau \cdot \frac{I}{T} \tag{19}$$

One can also unmask positions one by one, i.e. $K(\tau) = I - \tau$, sacrificing potential improvements in decoding speed.

Table 1: Translation results on the WMT 2016 Romanian→English task on the test set (`newstest2016`). (I = target hypothesis length, T = number of iterations for the hypothesis generation/refinements)

Category	Decoding Complexity	Method	BLEU [%] Length Predicted	BLEU [%] Length Gold
Non-autoregressive	$O(1)$	Fertility-based (Gu et al., 2018)	29.1	-
		CTC (Libovický & Helcl, 2018)	24.7	-
		Imitation learning (Wei et al., 2019)	28.9	-
		Reinforcement learning (Shao et al., 2019)	27.9	-
		Generative flow (Ma et al., 2019)	30.2	-
Non-autoregressive + Refinements	$O(T)$	Extra refinement model (Lee et al., 2018)	30.2	31.3
		MTM (This Work)	**30.5**	**32.0**
Autoregressive	$O(I)$	Enc-Dec Transformer	32.9	-

4 EXPERIMENTS

We implemented the MTM in the RETURNN framework (Doetsch et al., 2017) and evaluate the performance on the WMT 2016 Romanian→English translation task[1]. All data used in the experiments are preprocessed using the MOSES (Koehn et al., 2007) tokenizer and frequent-casing (Vilar et al., 2010). We learn a joint source/target byte pair encoding (BPE) (Sennrich et al., 2016) with 20k merge operations on the bilingual training data. Unless mentioned otherwise, we report results on the `newstest2016` test set, computed with case sensitivity and tokenization using the software SacreBLEU[2] (Post, 2018).

The MTMs in our experiments follow the base configuration of Vaswani et al. (2017), however, with a depth of 12 layers and 16 attention heads. They are trained using Adam (Kingma & Ba, 2015) with an initial learning rate of 0.0001 and a batch size of 7,200 tokens. Dropout is applied with a probability of 0.1 to all hidden layers and word embeddings. We set a checkpoint after every 400k sentence pairs of training data and reduce the learning rate by a factor of 0.7 whenever perplexity on the development set (`newsdev2016`) does not improve for nine consecutive checkpoints. The final models are selected after 200 checkpoints based on the development set perplexity.

During training, we select a certain percentage ρ_{s} of random target tokens to be corrupted. This parameter is selected randomly from a uniform distribution $\rho_{\text{s}} \sim \mathcal{U}[0, 1]$ in every training step. We further deviate from Devlin et al. (2019), by selecting the hyperparameters for corruption to be $\rho_{\text{mask}} = 0.6$, $\rho_{\text{rand}} = 0.3$, and $\rho_{\text{keep}} = 0.1$, which performed best for MTMs in our preliminary experiments.

The main results of the MTM are given in Table 1 along with comparative baselines. In total, our MTM, despite its extremely simple architecture, outperforms comparable constant-time NMT methods which do not depend on sequence lengths. Compared to the conventional autoregressive baseline, the MTM falls behind by only -2.4% BLEU with a constant number of decoding steps and the lower model complexity. Furthermore, a control experiment using the gold length instead of a predicted length improves the results from the same MTM model by 1.5% BLEU. This result minimizes the gap between our MTM and a comparable encoder-decoder model down to 0.9% BLEU, while our model has the ability to improve the decoding speed without retraining by simply reducing the number of iterations, thus trading in performance against speed.

Note that all other methods shown in Table 1 are based on the encoder-decoder architecture, which is more sophisticated. Moreover, the performance of Gu et al. (2018), Lee et al. (2018), Wei et al. (2019) and Shao et al. (2019) relies heavily on knowledge distillation from a well-trained autoregressive model, which involves building an additional NMT model and translating the entire training data with that model. This causes a lot more effort and computation time in training, while the MTM requires no such complicated steps, and its training is entirely end-to-end.

[1] http://www.statmt.org/wmt16/translation-task.html

[2] SacreBLEU configuration string: BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.2.17

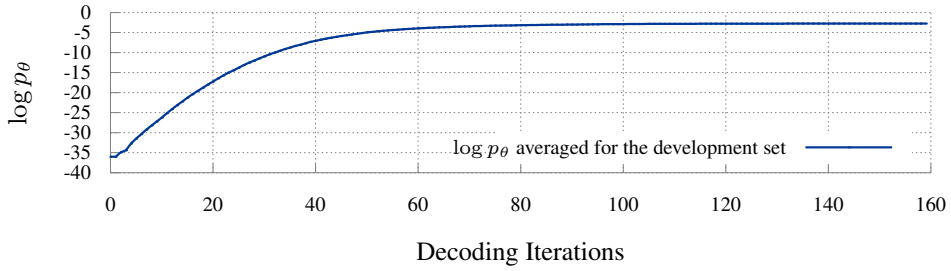

Figure 3: Average value of the log-probability over `newsdev2016` for each decoding iteration i.e. each refinement step in decoding with a maximum of 160 iterations.

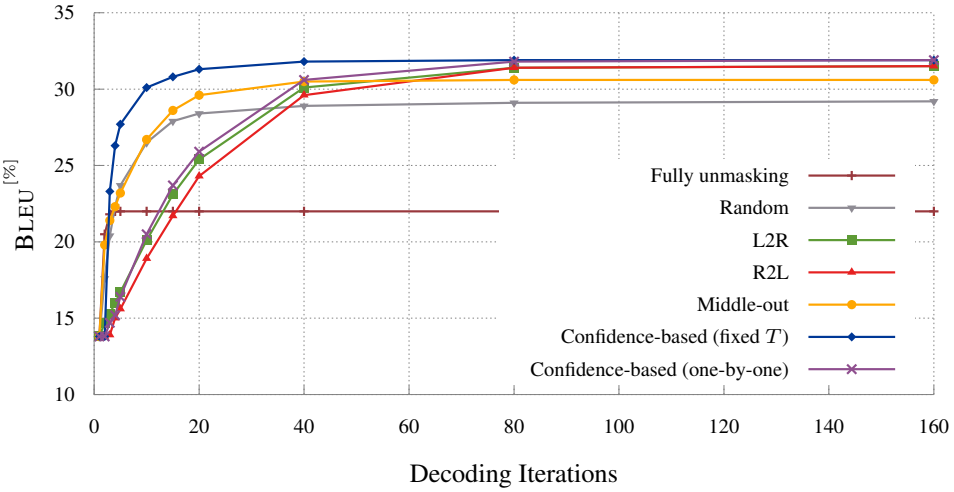

Figure 4: Comparison of different decoding strategies for the MTM for different amounts of decoding iterations T (on `newsdev2016`).

Ghazvininejad et al. (2019) demonstrate in their work that even better results could be possible by computing multiple translations in parallel for a set of most likely length hypotheses. This approach or even a beam-search variant of our iterative unmasking, will be a focus in our future work.

4.1 DECODING

As described in Section 3.3.1, the flexibility of the MTM allows us to easily implement different decoding strategies within a single model. Pure non-autoregressive decoding, i.e., a single forward pass to predict all target positions simultaneously, yields poor translation performance of 13.8% BLEU on the validation set (`newsdev2016`), which implies that several decoding iterations are needed to produce a good hypothesis. We know this to be true if the inequality in Equation (11) holds, i.e. if we see our model score improving in every iteration. Figure 3 shows that we can actually observe this property by monitoring the average model score throughout all iterations. Outputs for individual sentences might still worsen between two iterations. The overall score, however, shows a steady improvement in each iteration.

In Figure 4, we compare various decoding strategies and plot their performance for different number of decoding iterations T. "Fully unmasking", i.e. re-predicting all positions based on the previous hypothesis, improves the hypothesis fast in the early iterations but stagnates at 22% BLEU.

L2R, R2L, and confidence-based one-by-one all unmask one position at a time and show a very similar tendency with confidence-based one-by-one decoding reaching the strongest final performance of 31.9% BLEU. Confidence-based fixed-T unmasks several target positions per time step

and achieves similar performance. In contrast to position-wise unmasking, the decoding with a fixed number of T (and linear unmasking) only needs ten decoding iterations to reach close to optimal performance.

We test "middle-out" a variation of the L2R strategy to see whether the generation order is negligible as long as the sentence is generated contiguously. While this improves the hypothesis faster – most likely due to its doubled rate of unmasking – the final performance is worse than those of L2R or R2L decoding. Random selection of the decoding positions shows comparably fast improvements up to the 10th iterations, keeping up with middle-out, even though it reveals the hypothesis for a single position per iteration, however its performance saturates below most other decoding strategies.

Overall the best result for a low iteration count is obtained with confidence-based decoding with a fixed number of iterations. This shows that it is possible and sometimes even beneficial to hypothesize several positions simultaneously. We conclude that the choice of the decoding strategy has substantial impacts on the performance and hypothesize that a good decoding strategy relies on the model score to choose which target positions should be unmasked.

5 Conclusion

In this work we simplify the existing Transformer architecture by combining the traditional encoder and decoder elements into a single component. The resulting masked translation model is trained by concatenating source and target and applying BERT-style masking to the target sentence. The novel training strategy introduced with the MTM requires a rethinking of the search process and allows for various new decoding strategies to be applied in the theoretical framework we developed in this work. A detailed comparison shows that unmasking the sequence one-by-one gives the overall best performance, be it left-to-right, right-to-left, or confidence-based. Unveiling a constant number of tokens based on confidence in each decoding step, however, can achieve reasonable performance with a fixed, much smaller number of iterations.

We show that there is a potential of at least 1.5 % BLEU improvement that can be achieved by more elaborate length models, which yields itself as a good start for further research. Furthermore, we plan to extend the decoding strategies to work with beam search and verify our observations on further language pairs.

References

Parnia Bahar, Christopher Brix, and Hermann Ney. Towards two-dimensional sequence to sequence model in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3009–3015, 2018.

Dzmitry Bahdanau, KyunghyunF Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, 2018.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019. URL https://www.aclweb.org/anthology/N19-1423/.

Patrick Doetsch, Albert Zeyer, Paul Voigtlaender, Ilia Kulikov, Ralf Schlüter, and Hermann Ney. RETURNN: The RWTH extensible training framework for universal recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5345–5349. IEEE, 2017.

José AR Fonollosa, Noe Casas, and Marta R Costa-jussà. Joint source-target self attention with locality constraints. *arXiv preprint arXiv:1905.06596*, 2019.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):721–741, 1984.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Constant-time machine translation with conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 1631–1640, 2016.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. Modeling recurrence for transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*, pp. 7944–7954, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pp. 177–180. Association for Computational Linguistics, 2007.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*, 2018.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R Lyu, and Zhaopeng Tu. Information aggregation for multi-head attention with routing-by-agreement. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3566–3575, 2019a.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. Hint-based training for non-autoregressive machine translation. In *2019 Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*, Hong Kong, China, November 2019b.

Jindřich Libovickỳ and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3016–3021, 2018.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *2019 Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*, Hong Kong, China, November 2019.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 1715–1725, 2016.

Harshil Shah and David Barber. Generative neural machine translation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 1353–1362, 2018. URL http://papers.nips.cc/paper/7409-generative-neural-machine-translation.

Chenze Shao, Yang Feng, Jinchao Zhang, Fandong Meng, Xilin Chen, and Jie Zhou. Retrieving sequential information for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3013–3024, Florence, Italy, July 2019.

Yanyao Shen, Xu Tan, Di He, Tao Qin, and Tie-Yan Liu. Dense information flow for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1294–1303, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3104–3112, 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pp. 262–270, 2010.

Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094, 2019. URL http://arxiv.org/abs/1902.04094.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. Deep neural machine translation with linear associative unit. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 136–145, 2017.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. Imitation learning for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1304–1312, Florence, Italy, July 2019.

Baosong Yang, Longyue Wang, Derek Wong, Lidia S Chao, and Zhaopeng Tu. Convolutional self-attention networks. *arXiv preprint arXiv:1904.03107*, 2019.

Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 521–530, 2016.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383, 2016.

APPENDIX

In this section we present several ablation studies as well as a deeper look into the decoding to further investigate the MTM.

A LENGTH MODELING

Autoregressive models determine the output length by stopping the generation process once a special token that marks the end of the sentence (e.g. '</S>') is predicted. This approach is not compatible with the MTM decoding as target tokens for all positions are predicted in parallel. Therefore we assume a given output length I and a train length model $p(I|f_1^J)$ on the bilingual training data. In training, the true length is used, and in decoding, we choose $\arg\max_I p(I|f_1^J)$ as the target length. To model $p(I|f_1^J)$ and determine the impact of length model quality on the MTM, we test length models of different complexity:

- **Count-based Table**. For the unseen source lengths in training, we assume $I = J$.
- **Poisson Distribution**: a more smoothly parametrized model. For J's not appearing in the training data, we back off to a global distribution with the parameter λ that is learned via maximum likelihood overall source/target length pairs in the training data, i.e. $\{(I, J) \,|\, (f_1^J, e_1^I) \in \mathcal{D}\}$.
- **Recurrent Neural Network (RNN)**: We take the last hidden state as the input to a target length classifier, i.e. a linear projection with a softmax layer over the possible target lengths $I \in \{1, 2, ..., 200\}$.
- **Bidirectional RNN**: a variation of the above which employs a BiLSTM and uses the last hidden state of the forward and the backward LSTM for length prediction.

Table 2 verifies that the translation performance depends highly on the length model. We also report the averaged absolute difference of the predicted length and the reference length.

Table 2: Comparison of target hypothesis length modeling for the MTM on the development set (newsdev2016).

Length Model	Avg. ΔI	BLEU [%]
Count-based	0.70	30.4
Poisson distribution	0.83	31.5
RNN	0.53	31.7
Bidirectional RNN	0.23	31.9
Reference	-	33.5

The simplest count-based model shows the worst performance, where the target length is merely determined by a frequency table. The Poisson distribution models the length more systematically, giving +1.1% BLEU against the count-based model. RNN models consider not only the source length but also the whole sentence semantics, and slightly outperform the Poisson distribution. The bidirectional RNN model predicts the target length even better, but the translation performance only improves marginally. Note that using the Reference length improves even further by +1.6% BLEU over our strongest system, which shows that a good length model is a crucial component of the MTM system.

B MODEL HYPERPARAMETERS

As discussed earlier, the MTM architecture is equivalent to a traditional transformer encoder. Nevertheless, the way it is applied to the task at hand differs very much, even compared to an MLM. Thus it was crucial to do a thorough hyperparameter search to obtain an optimal model performance. The baseline MTM configuration we present here is already the product of many preliminary experiments, setting the number of attention heads $h = 16$, dropout $P_{\text{drop}} = 0.1$, and the learning rate reduction scheme. The ablation study presented in table 3 highlights the importance of both the

Table 3: Comparison for a selected set of hyperparameters: number of layers N, hidden size d_{model}, number of attention heads h, attention key size d_k, attention value size d_v, dropout probability P_{drop}

	N ($\Leftrightarrow N_{\text{enc}} + N_{\text{dec}}$)	d_{model}	h	d_k	d_v	P_{drop}	BLEU [%]	TER [%]
MTM	12	512	16	32	32	0.1	31.9	55.1
			4	128	128		31.6	55.5
			8	64	64		31.9	54.9
			32	16	16		31.3	55.8
	6						30.6	56.4
	18						32.0	54.9
	24						32.4	54.6
		256		16	16		29.4	57.8
		1024		64	64		32.2	54.8
						0.0	29.2	58.1
						0.2	31	56.1
Enc-Dec	6+6	512	8	64	64	0.2	34.7	52.3
	3+3						35.0	51.8
	9+9						35.7	51.4
	12+12						35.6	51.6
		256		32	32		33.6	53.1
		1024		128	128		35.5	51.5

number of attention heads and especially dropout. It also shows that it was crucial to increase the model depth N compared to the standard transformer encoder, by matching the total number of layers $N = 12$ as they are used in an encoder-decoder model.

C Derivation of Decoding

In this section, we show a detailed derivation of the decoding process which justifies our iterative optimization procedure and modularizes the unmasking procedure to apply the application of various decoding strategies.

Assuming we have a hypothesized target length $\hat{I}$, the goal is to find an optimal target sequence $\hat{E}$ given length $\hat{I}$ and source sentence F:

$$\hat{E}(F, \hat{I}) = \arg\max_{E} \{p(E|\hat{I}, F)\} \tag{20}$$

The MTM decoding to find such a sequence is performed in T iterations, whose intermediate hypotheses are introduced as latent variables $E^{(1)}, \ldots, E^{(T-1)}$:

$$= \arg\max_{E^{(T)}} \left\{ \sum_{E^{(1)}, \ldots, E^{(T-1)}} p(E^{(T)}, \ldots, E^{(1)}|F, \hat{I}) \right\} \tag{21}$$

where the last iteration should provide the final prediction, i.e. $E^{(T)} := E$. Applying the chain rule and a first-order Markov assumption on $E^{(\tau)}$ yields:

$$= \arg\max_{E^{(T)}} \left\{ \sum_{E^{(1)}, \ldots, E^{(T-1)}} \prod_{\tau=1}^{T} \left[p(E^{(\tau)}|E^{(\tau-1)}, F, \hat{I}) \right] \right\} \tag{22}$$

with $E^0 := \texttt{</M>}, \ldots, \texttt{</M>}$ a sequence of length $\hat{I}$. In a next step, we approximate the sum by a maximization and subsequently apply a logarithm to get:

$$\approx \arg\max_{E^{(T)}} \left\{ \max_{E^{(1)}, \ldots, E^{(T-1)}} \sum_{\tau=1}^{T} \underbrace{\left[\log p(E^{(\tau)}|E^{(\tau-1)}, F, \hat{I}) \right]}_{=:Q(E^{(\tau)}|E^{(\tau-1)}, F, \hat{I})} \right\} \tag{23}$$

To simplify further derivations, we introduce the score function Q here and do another approximation by considering only the score Q from a single maximum timestep instead of the full sum over $\tau = 1, \ldots, T$.

$$\approx \arg\max_{E^{(T)}} \left\{ \max_{E^{(1)},\ldots,E^{(T-1)}} \left[\max_{\tau=1,\ldots,T} Q(E^{(\tau)}|E^{(\tau-1)}, F, \hat{I}) \right] \right\} \tag{24}$$

Even with this approximation, we are still trying to find a value for each $E^{(\tau)}$ that optimizes the score of another iteration $\hat{\tau}$ via the connection of dependencies in Q. As this is impractical to compute, we alleviate the problem by focusing on a step-wise maximization. For this we define the sequence $\hat{E}^{(\tau)}$ (with $\hat{E}^{(0)} := E^{(0)}$) as the best hypothesis of iteration τ given $\hat{E}^{(\tau-1)}$, i.e.:

$$\hat{E}^{(\tau)} := \hat{E}^{(\tau)}(\hat{E}^{(\tau-1)}, F, \hat{I}) = \arg\max_{E^{(\tau)}} Q(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) \tag{25}$$

If we use $\hat{E}^{(\tau-1)}$ instead of $E^{(\tau-1)}$ as the dependency in Q, we restrict the optimization to maximizing each step independently, given its predecessors optimum:

$$\hat{E}(F, \hat{I}) \approx \arg\max_{E^{(T)}} \left\{ \max_{E^{(1)},\ldots,E^{(T-1)}} \left[\max_{\tau=1,\ldots,T} Q(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) \right] \right\} \tag{26}$$

Ideally, this iterative procedure should improve the score Q in each iteration, i.e.:

$$\max_{E^{(\tau+1)}} Q(E^{(\tau+1)}|\hat{E}^{(\tau)}, F, \hat{I}) \geq \max_{E^{(\tau)}} Q(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) \;\; \forall \tau = 1, \ldots, T-1 \tag{27}$$

While Equation (27) is not true for the general case we observe empirically that this the statement is true on average (see Figure 3). This means that the maximum score to be obtained in the last iteration T:

$$\hat{E}(F, \hat{I}) \approx \arg\max_{E^{(T)}} \left\{ \max_{E^{(1)},\ldots,E^{(T-1)}} \left[Q(E^{(T)}|\hat{E}^{(T-1)}, F, \hat{I}) \right] \right\} \tag{28}$$

which can be simplified to:

$$= \arg\max_{E^{(T)}} Q(E^{(T)}|\hat{E}^{(T-1)}, F, \hat{I}) = \hat{E}^{(T)} \tag{29}$$

Thus, in order to approximate the best hypothesis it is sufficient to follow the recursive definition presented in Equation (25), which can be computed straight forward resulting in an iterative decoding scheme.

To allow a more fine-grained control of the decoding process, we introduce a (partially) masked version of hypothesis $\hat{E}^{(\tau-1)}$ as a latent variable $\tilde{E}^{(\tau-1)}$ in the score computation:

$$Q(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) = \log p(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) \tag{30}$$

$$= \log \left[\sum_{\tilde{E}^{(\tau-1)}} p(E^{(\tau)}, \tilde{E}^{(\tau-1)}|\hat{E}^{(\tau-1)}, F, \hat{I}) \right] \tag{31}$$

$$= \log \left[\sum_{\tilde{E}^{(\tau-1)}} p_\theta(E^{(\tau)}|\tilde{E}^{(\tau-1)}, F, \hat{I}) \cdot p_{\mathrm{m}}(\tilde{E}^{(\tau-1)}|\hat{E}^{(\tau-1)}, \hat{I}) \right] \tag{32}$$

where the probability of $E^{(\tau)}$ is modeled by the neural network with parameters θ and the masked sequence $\tilde{E}^{(\tau-1)}$ is given by p_{m}, which defines a specific search strategy (Section 3.3.1). Note here that, to remove redundant information in the modeling, p_θ drops the direct dependency on $\hat{E}^{(\tau-1)}$ and p_{m} ignores the source sentence F. As the last simplifying step, the intractable sum over the masked sequence $\tilde{E}^{(\tau-1)}$ is replaced by a point sample, i.e. $\tilde{E}^{(\tau-1)} \sim p_{\mathrm{m}}(\tilde{E}^{(\tau-1)}|\hat{E}^{(\tau-1)})$ for each $\tau = 1, \ldots, T$. This may even provide the exact result if p_m is designed for a deterministic search strategy.

$$Q(E^{(\tau)}|\hat{E}^{(\tau-1)}, F, \hat{I}) = \log p_\theta(E^{(\tau)}|\tilde{E}^{(\tau-1)}, F, \hat{I}) + \log p_{\mathrm{m}}(\tilde{E}^{(\tau-1)}|\hat{E}^{(\tau-1)}, \hat{I}) \tag{33}$$

$$= \log p_\theta(E^{(\tau)}|\tilde{E}^{(\tau-1)}, F, \hat{I}) \tag{34}$$

D DECODING STRATEGIES

For completeness we report the strongest result for each decoding strategy from Figure 4 in Table 4.

Table 4: Comparison of different decoding strategies in greedy decoding for the MTM on the development set (`newsdev2016`) of the Romanian→English task.

Category	Decoding Complexity	Decoding Strategy	BLEU [%]
Non-autoregressive	$O(1)$	Predict all positions	13.8
Non-autoregressive + Refinements	$O(T)$	Fully unmasking	22.0
		Confidence-based (fixed T)	31.9
	$O(I)$	Confidence-based (one-by-one)	31.9
Autoregressive	$O(I)$	Left-to-right	31.5
		Right-to-left	31.5
		Middle-out	30.6
		Random	29.2