Exploring Unsupervised Pretraining Objectives for Machine Translation

Christos Baziotis, Ivan Titov, Alexandra Birch and **Barry Haddow**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
c.baziotis@ed.ac.uk, ititov@inf.ed.ac.uk,
a.birch@ed.ac.uk, bhaddow@inf.ed.ac.uk

Abstract

Unsupervised cross-lingual pretraining has achieved strong results in neural machine translation (NMT), by drastically reducing the need for large parallel data. Most approaches adapt masked-language modeling (MLM) to sequence-to-sequence architectures, by masking parts of the input and reconstructing them in the decoder. In this work, we systematically compare masking with alternative objectives that produce inputs resembling real (full) sentences, by reordering and replacing words based on their context. We pretrain models with different methods on English↔German, English↔Nepali and English↔Sinhala monolingual data, and evaluate them on NMT. In (semi-) supervised NMT, varying the pretraining objective leads to surprisingly small differences in the finetuned performance, whereas unsupervised NMT is much more sensitive to it. To understand these results, we thoroughly study the pretrained models and verify that they encode and use information in different ways. We conclude that finetuning on *parallel* data is mostly sensitive to few properties that are shared by most models, such as a strong decoder, in contrast to unsupervised NMT that also requires models with strong cross-lingual abilities.

1 Introduction

Neural machine translation (NMT) is notoriously data-hungry (Koehn and Knowles, 2017). To learn a strong model it requires large, high-quality and in-domain parallel data, which exist only for a few language-pairs. The most successful approach for improving low-resource NMT is backtranslation (Sennrich et al., 2016), that exploits abundant monolingual corpora to augment the parallel with synthetic data. However, in low-resource settings, it may fail to improve or even degrade translation quality if the initial model is not strong enough (Imankulova et al., 2017; Burlot and Yvon, 2018).

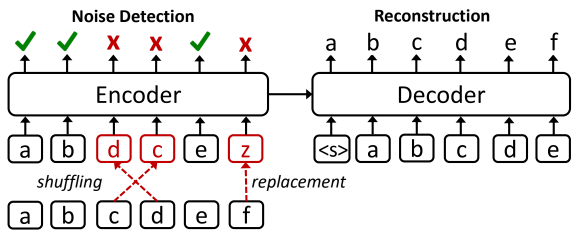


Figure 1: We consider noising methods that produce inputs which resemble real sentences, unlike masking.

Unsupervised pretraining is a complementary technique, that has revolutionized many natural language understanding (NLU) tasks (Wang et al., 2019). The dominant approach is to train a (large) model on a lot of unlabeled data using the masked language modeling (MLM; Devlin et al. (2019)) objective and then finetune it on a downstream task. Besides improving generalization, good initialization drastically reduces the need for labelled data. This paradigm has been applied recently to NMT yielding impressive results in low-resource settings, with models such as XLM (Conneau and Lample, 2019), MASS (Song et al., 2019) and BART/mBART (Lewis et al., 2020b; Liu et al., 2020), that adapt MLM to sequence-to-sequence architectures. Although pretraining alone is not enough to outperform backtranslation, it helps the initial model to produce synthetic data of sufficient quality, and combining them yields further improvements.

Most prior work in pretraining has focused on optimizing the masking strategy (Rogers et al., 2021). Similarly, MASS and mBART consider slightly different masking strategies. However, due to differences in their experimental setup (i.e., capacity or training data) and lack of analysis that goes beyond evaluation on downstream tasks, it is unclear if there is a meaningful difference between them, as far as NMT is concerned. They also suffer from a pretraining-finetuning discrepancy (Yang et al., 2019), in which a model is pretrained on masked inputs, but finetuned on full sentences.

In this work[1], we explore different objectives to masking for unsupervised cross-lingual pretraining. We inject noise that creates examples (Fig. 1), similar to those encountered in finetuning, unlike masking. This includes, randomly replacing input words based on their context using a cross-lingual generator, inspired by Clark et al. (2020), and locally reordering input words, which prevents the cross-attention from naively (monotonically) attending over the source. We also explore auxiliary losses over the encoder to improve its representations.

First, we pretrain models with different configurations, on English-German, English-Nepali and English-Sinhala monolingual data. Then, we *systematically* compare them on the downstream tasks of supervised, semi-supervised and unsupervised NMT. In (semi-) supervised NMT, we observe that models yield surprisingly similar results, although some methods are better than others. We find that even pretraining with shuffled inputs leads to significant improvements over random initialization, similar to the concurrent work of Sinha et al. (2021) on pretrained encoders for NLU. Unsupervised NMT, however, reveals large (up to 9 BLEU points) differences, and against our expectations, masking achieves the best performance. To understand these results, unlike prior work, we thoroughly analyze the pretrained models using a series of probes, and discover that each objective drives the models to encode and use information in unique ways.

Based on our findings, we conclude that each finetuning process is sensitive to specific properties of pretrained models, similar to Artetxe et al. (2020). We hypothesize that (semi-) supervised NMT is mostly sensitive to the LM abilities of pretrained models, as the source$\rightarrow$target mappings can be learnt from the parallel data. Unsupervised NMT requires models to also rely on their own word-translation abilities. Our contributions are:

1. We *systematically* compare many pretraining methods, including alternatives to masking, in three NMT tasks and for three language-pairs.
2. We discover that (semi-) supervised NMT is not sensitive to the pretraining strategies. Our ablation (§4.4) suggests that a strong decoder is the most important factor, while differences in the encoder (§4.5) don't affect the results.
3. Unsupervised setting is much more sensitive to the pretraining objective, and masking methods are the most effective. We hypothesise that learning to copy is important here (§5.2) as is cross-lingual encoding (§4.5).
4. We analyze the pretrained models with a series of probes (§5.1, §5.2, §5.3), and show noticeable differences in how they encode and use information, offering valuable insights.

2 Related Work

Pretraining for NMT Ramachandran et al. (2017) first explored unsupervised pretraining for NMT using LMs trained on monolingual data of the source and target languages to initialize the encoder and decoder of an RNN-based TM (Bahdanau et al., 2015). Conneau and Lample (2019) adopt the same approach, by extending BERT/MLM (Devlin et al., 2019) to the cross-lingual setting (XLM). They randomly mask tokens from input sentences in many languages, and the model is trained to predict them. However, the same pretrained XLM is used to (separately) initialize both the encoder and decoder of a downstream translation model (TM), which neglects the interaction between them. MASS (Song et al., 2019) addresses this limitation, by extending MLM to sequence-to-sequence pretraining, which includes the cross-attention mechanism, and achieved further improvements in low-resource and unsupervised NMT. Liu et al. (2020), concurrently demonstrated comparable results with a similar approach (mBART), but on a larger scale. Both mBART and MASS, consider different strategies for reconstructing masked input spans.

Objectives Yang et al. (2019) point out that BERT (Devlin et al., 2019) is pretrained with masked inputs, but then finetuned on full sentences, which creates a discrepancy. To address this, they change the self-attention in Transformers (Vaswani et al., 2017) to predict tokens conditioned on all permutations of other tokens in a sentence and Song et al. (2020) extend this to sequence-level pretraining for NLU. MARGE (Lewis et al., 2020a) explores multi-lingual pretraining for document-level NMT, by reconstructing texts from a set of retrieved relevant documents. Clark et al. (2020) propose the replaced token detection (RTD) objective for pretraining text encoders. They replace tokens with samples from a MLM and train the encoder as a discriminator to predict whether each word is real or fake. Similar ideas have been previously explored in NMT with contextual data augmentation (Fadaee et al., 2017; Kobayashi, 2018; Gao et al., 2019).

[1]Code at github.com/cbaziotis/nmt-pretraining-objectives

3 Pretraining

Our pretraining model is a multilingual denoising sequence autoencoder, based on the Transformer (Vaswani et al., 2017).

We assume access to a corpus of unpaired data, containing text in two languages A, B. Given a text sequence of N tokens $\boldsymbol{x} = \langle x_1, x_2, ..., x_N \rangle$ we first add noise to it and obtain its corrupted version $\boldsymbol{x'}$. An encoder transforms $\boldsymbol{x'}$ into a sequence of contextualized representations $h(\boldsymbol{x'}) = \langle h_1, h_2, ..., h_N \rangle$, which are given as input to the decoder, that produces a reconstruction of $\boldsymbol{x}$. The reconstruction loss is the negative log-likelihood (NLL) of $\boldsymbol{x}$:

$$\mathcal{L}_{\text{R}} = \frac{1}{N} \sum_{t=1}^{N} - \log p(\boldsymbol{x}_t | \boldsymbol{x}_{<t}, h(\boldsymbol{x'})) \quad (1)$$

Each batch contains sentences in either A, or B and to distinguish between them, we add language id tokens at the end of the source sentences, and the beginning of the target sentences.

3.1 Pretraining Methods

In this section, we describe the methods that we use to inject noise into the model. We also explore auxiliary losses over the encoder, aiming to improve the input representations.

Masking Similar to prior work, we replace a random subsequence M of the input tokens with a special `[MASK]` token and train the model to reconstruct the original input. We consider masking words as well as spans following mBART.

Masking + eMLM When using masking noise we also explore the addition of an auxiliary MLM loss over the *encoder* to which we will refer as eMLM. This explicitly trains the encoder to reconstruct the representations of masked tokens:

$$\mathcal{L}_{\text{eMLM}} = \frac{1}{|M|} \sum_{t \in M} - \log p(\boldsymbol{x}_t | \boldsymbol{x}_{\notin M})) \quad (2)$$

Replacing We inject word replacement noise, by extending Clark et al. (2020) to the cross-lingual setting. Specifically, we jointly train a *separate* cross-lingual (mBERT-like) MLM generator. First, we mask a random subset M of the input tokens[2]and the generator is trained to predict them:

$$\mathcal{L}_{\text{G}} = \frac{1}{|M|} \sum_{t \in M} - \log p(\boldsymbol{x}_t | \boldsymbol{x}_{\notin M})) \quad (3)$$

For each masked token $\boldsymbol{x}_t$, the generator produces a distribution $p_{\text{G}}(\boldsymbol{x}_t | \boldsymbol{x}_{\neq t})$. We replace the masked tokens with samples from p_{G} to obtain $\boldsymbol{x'}$, and feed the updated (corrupted) input to the encoder. We consider two configurations for the generator:

- *Untied*: The default setting, we use a small generator, which is half the size (x0.5 parameters) of the encoder, following Clark et al. (2020).
- *Tied*: We tie the weights of the encoder and the generator. This setting implicitly adds an auxiliary MLM loss over the encoder, and can be thought as a counterpart of "replace+eMLM".

Replacement + RTD Motivated by the results of Clark et al. (2020) in monolingual NLU, we add a replacement token detection (RTD) head over the encoder that gives direct supervision to the model regarding the location of noise. The RTD head $D(\cdot)$ is a token-level discriminator over the encoder outputs $h(\boldsymbol{x'})$, that predicts if an input token $\boldsymbol{x'_t}$ is original or replaced. We parameterize D with a non-linear projection followed by a sigmoid function: $D(\boldsymbol{x'_t}) = \text{sigmoid}(\boldsymbol{u}^\top \text{RELU}(\boldsymbol{W_D}\, h(\boldsymbol{x'_t})))$. The RTD loss is defined as the average token-level binary cross-entropy:

$$\mathcal{L}_{\text{RTD}} = \frac{1}{N} \sum_{t=1}^{N} - (\boldsymbol{x}'_t = \text{original}) \log D(\boldsymbol{x}'_t) \quad (4)$$
$$- (\boldsymbol{x}'_t \neq \text{original})(1 - D(\boldsymbol{x}'_t))$$

Shuffling Although pretrained models, such as MASS or mBART, do pretrain the cross-attention mechanism, the input words remain in their original positions and this biases the models into learning only naive monotonic alignments. To *actively* pretrain the cross-attention, we locally shuffle a random subset of *whole-words* in the input, using the method of (Lample et al., 2018). The length of reordering is bounded by k (positions). Small k introduce local shuffling, while large k allow words to be moved farther from their original position, making the input more like a bag-of-words (BoW).

3.2 Optimization

During pretraining, we minimize a weighted sum of the reconstruction L_{R}, and depending on the method, some of the auxiliary L_{eMLM}, L_{G}, L_{RTD} losses. We assign equal weight ($\lambda = 1$) to all losses, except for L_{RTD} for which we set its weight $\lambda = 25$ following Clark et al. (2020), to account for the fact that is in a different scale.

[2]We use whole-word masking, that masks all the (subword) tokens of a word, instead of independent token masking.

Method	en→de		de→en		en→ne	ne→en	en→si	si→en
	wmt18	wmt19	wmt18	wmt19				
random	26.2$_{\pm0.1}$	25.3$_{\pm0.1}$	27.6$_{\pm0.1}$	19.1$_{\pm0.3}$	3.3$_{\pm0.1}$	6.5$_{\pm0.1}$	2.5$_{\pm0.1}$	6.5$_{\pm0.1}$
mask=35%	33.3$_{\pm0.1}$	30.7$_{\pm0.2}$	33.2$_{\pm0.0}$	**25.4**$_{\pm0.0}$	5.1$_{\pm0.1}$	10.2$_{\pm0.1}$	3.7$_{\pm0.0}$	10.0$_{\pm0.1}$
mask=35% +eMLM	33.4$_{\pm0.0}$	30.6$_{\pm0.1}$	33.5$_{\pm0.1}$	25.2$_{\pm0.2}$	**5.3**$_{\pm0.0}$	**10.8**$_{\pm0.1}$	**4.0**$_{\pm0.0}$	10.4$_{\pm0.1}$
mask=35% (span)	33.3$_{\pm0.1}$	30.5$_{\pm0.1}$	33.4$_{\pm0.0}$	25.2$_{\pm0.0}$	5.1$_{\pm0.1}$	10.1$_{\pm0.1}$	3.9$_{\pm0.1}$	9.9$_{\pm0.1}$
shuffle=5	31.6$_{\pm0.1}$	28.7$_{\pm0.0}$	31.7$_{\pm0.0}$	23.9$_{\pm0.1}$	4.9$_{\pm0.0}$	9.9$_{\pm0.1}$	3.4$_{\pm0.0}$	10.1$_{\pm0.1}$
replace=35%	33.9$_{\pm0.0}$	30.3$_{\pm0.2}$	33.5$_{\pm0.1}$	**25.4**$_{\pm0.1}$	5.1$_{\pm0.1}$	9.9$_{\pm0.1}$	3.7$_{\pm0.0}$	9.8$_{\pm0.0}$
replace=35% +RTD	32.9$_{\pm0.1}$	30.0$_{\pm0.0}$	32.5$_{\pm0.0}$	24.4$_{\pm0.1}$	5.0$_{\pm0.0}$	9.9$_{\pm0.1}$	3.4$_{\pm0.1}$	9.7$_{\pm0.2}$
replace=35% +tied	**34.2**$_{\pm0.0}$	**30.8**$_{\pm0.1}$	**33.7**$_{\pm0.1}$	25.3$_{\pm0.2}$	**5.3**$_{\pm0.0}$	10.6$_{\pm0.1}$	3.7$_{\pm0.0}$	**10.5**$_{\pm0.1}$
+ shuffle=3	34.0$_{\pm0.0}$	$\underline{31.1}$$_{\pm0.1}$	33.4$_{\pm0.1}$	25.1$_{\pm0.2}$	$\underline{5.5}$$_{\pm0.0}$	$\underline{11.0}$$_{\pm0.0}$	$\underline{4.0}$$_{\pm0.0}$	$\underline{10.8}$$_{\pm0.1}$

Table 1: Supervised NMT results. We report the average of 3 runs and the standard error of the mean (SEM).

4 Experiments

Datasets We focus on low-resource translation and consider three diverse language-pairs: English-German, English-Nepali and English-Sinhala. For English-German, we use the low-resource WMT News Commentary v13 (Bojar et al., 2018) [3] parallel dataset, which contains approximately 275K sentences. For pretraining, we use as monolingual data the WMT News Crawl articles (Bojar et al., 2018) from the year 2007 to 2017, which comprise 190M and 270M sentences for English and German, respectively. For English-Nepali and English-Sinhala, we use the same data as in Guzmán et al. (2019). The (pretraining) monolingual data contain 5M sentences from Common Crawl and Wikipedia per language, while the parallel data are approximately 600K sentences from the Bible, Open Subtitles, GNOME/KDE/Ubuntu, and Paracrawl.

Pre-processing For Nepali and Sinhala, we use the preprocessing scripts[4] provided by Guzmán et al. (2019), whereas for English and German, we use directly the raw data without any preprocessing. We use sentencepiece (SPM; Kudo and Richardson (2018)) with the "unigram" model, to train a subword-unit tokenization model on the concatenation of the monolingual data of each language-pair. We learn a joint vocabulary of 60K symbols for the English-German models, and 20K symbols for the English-Nepali and English-Sinhala models.

Evaluation For English-German, we use the WMT *newstest2017* as dev-set and the *newstest2018* and *newstest2019* as test-sets. For English-Nepali and English-Sinhala, we use the evaluation datasets provided by Guzmán et al. (2019), which are drawn from Wikipedia articles. We evaluate models using BLEU (Papineni et al., 2002) computed with SacreBLEU (Post, 2018). We report detokenized BLEU when translating into German and English, and tokenized BLEU when translating into Nepali and Sinhala, following Guzmán et al. (2019). At test time, we decode with beam search, using beams of size 5.

Model and Training Our models are based on the Transformer architecture (Vaswani et al., 2017). We use the Transformer-base configuration to reduce the computational cost and be able to explore more methods. We describe in detail the model architecture and hyperparameters, as well as the pretraining and finetuning processes, in Appendix §A. Our code is based on the official mBART implementation in Fairseq (Ott et al., 2019).

4.1 Supervised Translation

In our first experiment (Table 1), we evaluate each pretrained model on supervised NMT by finetuning it on the parallel data. As a baseline we use a randomly initialized TM with identical configuration and vocabulary to that of the pretrained models, denoted as "random". We also pretrain a model equivalent to mBART denoted as "mask (span)".

Results All pretraining methods yield large improvements over random initialization, in all directions and language pairs. The differences between each method are more pronounced in en↔de, whereas in en↔ne and en↔si, all models reach much lower scores, especially in en→X, probably because of low-quality training data, and a domain mismatch between the parallel and test data.

When we compare each type of input noise in isolation, we observe that masking and replacement achieve similar results, and both are better than shuffling. However, pretraining with shuffling noise alone still yields surprisingly strong results. It improves over random initialization in all experiments, and it even reaches similar BLEU scores to masking and replacements in en↔ne and en↔si. Note that, the common denominator in all pretrain-

[3] http://www.statmt.org/wmt18/translation-task.html
[4] https://github.com/facebookresearch/flores

ing methods is the decoder, which is trained as a conditional LM with teacher forcing and the only difference is the input conditioning context. This implies that *one* key factor in pretraining for NMT is improving the LM capabilities of the decoder.

Auxiliary Losses The best results are obtained by "mask+eMLM" and "replace+tied", both of which benefit from an encoder MLM loss. We observe, that eMLM and tying are more effective in the X$\rightarrow$en than en$\rightarrow$X direction, especially for en$\leftrightarrow$ne and en$\leftrightarrow$si. This makes intuitive sense because eMLM improves the representations of the encoder, which is more important for languages with limited or low-quality data. Specifically, we observe that adding eMLM improves BLEU by +0.7 for ne$\rightarrow$en, +0.4 for si$\rightarrow$en and tying the generator with the encoder yields +0.7 BLEU for ne$\rightarrow$en, +0.7 BLEU for si$\rightarrow$en. Wang et al. (2020) make a similar observation in experiments in multilingual NMT.

Incorporating RTD, however, has a negative effect in most experiments. This is unexpected, given that Clark et al. (2020) showed that pretraining text encoders with RTD outperformed MLM in NLU tasks. This warns us that methods which produce strong encoders for NLU might not necessarily improve encoders for NMT. Note that, Siddhant et al. (2020); Wang et al. (2020) have discovered similar surprising results in cross-lingual NLU tasks.

Noise Combination We also consider a combination of the best replacement-based variant "replace+tied" with shuffling. Shuffling is applied after the replacements are sampled and we limit the length of reordering to $k = 3$ to prevent extreme corruption of the input. We observe that this combination yields small gains in most experiments. We also explored more pretraining methods and configurations, including the injection of noise into the decoder, but they didn't produce significant differences in terms of BLEU (see Appendix D).

4.1.1 Parameter Sensitivity Analysis

We also explore how changing key parameters of each pretraining method affects performance in supervised NMT. We report the results in Table 2. We observe there is a "sweet-spot" for the amount of noise used in each method. Shuffling shows larger variability and we find that by making the token swaps less local (i.e., increasing k that makes the input more BoW), yields better results.

Generator Size Next, we focus on why tying the encoder and generator yields better results. Either the encoder benefits from the implicit MLM loss, or tying improves the generator and consequently its samples. We train an *untied* model with equal capacity to the encoder "replace=35% (x1.0)", and a similar model, but we sample replacements with nucleus sampling[5] (Holtzman et al., 2020) with top-p=0.9, to avoid low-probability tokens. Neither of those variants yields any measurable difference with "replace=35%", which suggests that MLM is responsible for the improvements.

Method	**en→de**		**de→en**	
	wmt18	wmt19	wmt18	wmt19
mask=15%	32.7±0.1	30.4±0.1	32.9±0.0	25.1±0.1
mask=35%	**33.3**±0.1	**30.7**±0.2	**33.2**±0.0	**25.4**±0.0
mask=50%	33.2±0.0	30.4±0.0	33.1±0.1	25.2±0.1
shuffle=3	30.3±0.1	27.9±0.0	30.5±0.0	22.9±0.1
shuffle=5	**31.6**±0.1	**28.7**±0.0	**31.7**±0.0	**23.9**±0.1
replace=15%	33.8±0.1	**30.4**±0.1	33.1±0.1	25.3±0.2
replace=35%	**33.9**±0.0	30.3±0.2	**33.5**±0.1	**25.4**±0.1
replace=50%	33.3±0.0	30.3±0.1	32.8±0.0	24.7±0.2
replace=35% (1.0x)	33.7±0.1	30.6±0.0	33.3±0.0	25.0±0.1
replace=35% +nucleus	**33.9**±0.0	**30.9**±0.1	**33.6**±0.0	**25.3**±0.0
replace=35% +RTD=4	**33.2**±0.0	29.8±0.1	32.4±0.1	**24.5**±0.0
replace=35% +RTD=6	32.9±0.1	**30.0**±0.0	**32.5**±0.0	24.4±0.1

Table 2: Supervised NMT results (mean and SEM of 3 runs), for different configurations of pretrained models.

RTD Position We also explore if the position of the RTD head is responsible for its negative effects in NMT, as it might force the encoder to preserve information irrelevant for NMT to its outputs. To test this, we train a model with RTD over its fourth (RTD=4) instead of last/top (RTD=6) layer. We find that this change has a marginal effect on BLEU.

4.2 Semi-supervised Translation

In Table 3 we report results for semi-supervised NMT, using backtranslation. Both the forward and backward TM are initialized from the same model. We generate the backtranslations from the same monolingual data that we used for pretraining with greedy-sampling, which is preferable for weak TMs (Edunov et al., 2018), and upsample the real data to maintain a 1:1 ratio with the synthetic data.

Results Backtranslation yields significant gains in all experiments, and the initialization from pretrained models boosts performance even more. The relevant performance between methods is consistent with the supervised NMT, but their differences shrink further. We suspect that adding more data narrows the room for improvement of pretraining.

One exception is the initialization from "shuffle=5", which yields marginal gains in en$\leftrightarrow$de and even fails to reach the randomly initialized model

Method	en→de		de→en		en→ne	ne→en	en→si	si→en
	wmt18	wmt19	wmt18	wmt19				
random	34.8	29.2	37.4	24.8	5.8	13.9	6.5	12.9
mask=35%	38.3	31.5	38.8	27.7	6.3	14.9	6.5	13.5
mask=35% +eMLM	38.4	**31.8**	**39.0**	27.4	**6.6**	15.0	**7.3**	**14.2**
mask=35% (span)	38.3	31.6	39.0	27.5	6.4	14.4	6.3	13.4
shuffle=5	36.2	30.8	36.9	26.1	6.4	13.5	6.3	11.9
replace=35%	38.2	31.2	38.6	27.5	6.4	14.7	6.0	13.1
replace=35% +RTD	37.7	31.3	38.2	27.5	6.1	14.2	5.9	12.8
replace=35% +tied	**38.5**	31.7	38.8	**27.8**	6.4	**15.1**	6.6	13.7
+shuffle=3	38.2	31.6	38.6	27.0	6.5	15.4	7.4	14.2

Table 3: Finetuning results to **semisupervised** NMT. Each TM is trained on the concatenation of real and back-translated sentences, obtained by a backward TM initialized from the same pretrained model.

Method	en→de		de→en	
	wmt18	wmt19	wmt18	wmt19
mask=35%	**25.6**	**19.1**	**29.3**	**20.3**
mask=35% +eMLM	25.2	18.7	28.9	19.8
mask=35% (span)	24.7	18.1	28.3	19.7
shuffle=5	17.1	13.1	20.5	15.6
replace=35%	23.3	17.5	27.4	19.4
replace=35% +RTD	22.8	16.8	26.5	18.5
replace=35% +tied	24.1	17.8	27.8	19.3

Table 4: Finetuning results to **unsupervised** NMT.

in si↔en and ne↔en. Note that, one of the advantages of backtranslation is improving the LM capabilities of the decoder with the addition of more clean target data. We hypothesize that shuffling noise mainly pretrains the decoder as a LM, and its benefits are largely neutralized by backtranslation.

4.3 Unsupervised Translation

In Table 4 we evaluate the pretrained models on unsupervised NMT (Artetxe et al., 2018; Lample et al., 2018), using *only* the monolingual data. In this experiment we focus on en↔de, because unsupervised NMT in en↔ne and en↔si yields very low BLEU scores (Guzmán et al., 2019; Liu et al., 2020). In each batch, we generate backtranslations on-the-fly in the target language and the model is optimized to reconstruct the original sentences (i.e., en→de′→$\hat{\text{en}}$), following the same finetuning process as mBART (see Appendix B for details).

Results Unlike the experiments on parallel data (§4.1, §4.2), unsupervised NMT reveals large differences between pretrained models. Strikingly, "shuffle=5" yields the lowest BLEU scores by a large margin. This further supports the hypothesis that its primary strength is its decoder. We hypothesize that all pretraining methods produce strong decoders, but not encoders. Since in supervised NMT the models can learn source-to-target mappings from the parallel data, models with better cross-lingual abilities have a small edge. However, in the unsupervised setting having a fluent decoder alone is not enough as the models have to rely on their own word-translation capabilities (§4.5) to be able to produce sufficient backtranslations.

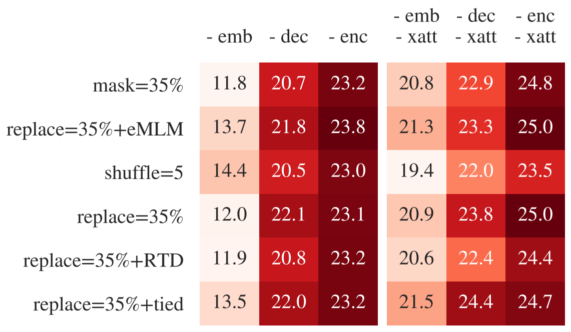
Another unexpected result is that pretraining with masked inputs outperforms replacements. Replacement-based models should intuitively be favoured, because the mistakes injected by the generator during pretraining resemble those produced by backtranslation. Masking-based models, however, are exposed to very different inputs without any signal from parallel data to help them transition to the new training regime, unlike supervised NMT.

In our analysis (§5.2), we find that masking biases models towards copying from the input, whereas replacements make models more "cautious" because some input words are fake. We hypothesize that the ability of mask-based models to copy words, such as dates or named entities, is critical to kickstart the backtranslation process.

4.4 Supervised Translation Ablations

To estimate how important of each part of the pretrained model is for NMT, we conduct an ablation experiment. First, we transfer all the weights of a pretrained model to a downstream model, except the weights of the ablated component. Next, we freeze the pretrained weights for all components except for the ablated one, which we reinitialise randomly. Finally we finetune only the ablated component, to isolate the effects on the final score to the component and prevent the other components from compensating (details in Appendix B.1).

We divide the model into four parts: (1) the *embedding* matrix, which is used for both the encoder and decoder embeddings and the vocabulary (output) projection, (2) the *encoder* layers, (3) the *decoder* layers and (4) the *cross-attention* layers, which link the decoder with the encoder. We re-



	- emb	- dec	- enc	- emb - xatt	- dec - xatt	- enc - xatt
mask=35%	11.8	20.7	23.2	20.8	22.9	24.8
replace=35%+eMLM	13.7	21.8	23.8	21.3	23.3	25.0
shuffle=5	14.4	20.5	23.0	19.4	22.0	23.5
replace=35%	12.0	22.1	23.1	20.9	23.8	25.0
replace=35%+RTD	11.9	20.8	23.2	20.6	22.4	24.4
replace=35%+tied	13.5	22.0	23.2	21.5	24.4	24.7

Figure 2: Ablation results for supervised NMT on de→en (wmt19). We reset each main component individually (left) and with the cross-attention (right).

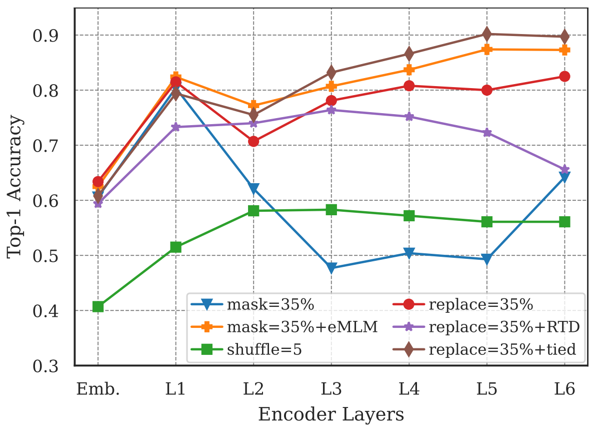


Figure 3: Parallel sentence retrieval accuracy (de→en).

port the de→en supervised NMT ablation results in Figure 2. Higher BLEU scores show that a model can better recover after an ablation, which we interpret as an indication that the ablated parameters are less important. We ablate each component both individually and combined with cross-attention, to allow the model to better learn to connect source and target representations. We find in both settings, that all models recover better after resetting their encoders than their decoders, implying that the pretrained decoder parameters are more important.

4.5 Cross-lingual Sentence Retrieval

To study the cross-lingual abilities of the encoder of each pretrained model, we evaluate them on parallel sentence retrieval. In Figure 3, we report the (per layer) accuracy of each encoder, on the de→en Tatoeba test set (Tiedemann, 2020)[6].

The results indicate that different objectives do affect the encoder's cross-lingual abilities. The fact that this is not reflected in the BLEU scores of the finetuned models further supports that even a small parallel data-set is enough to align them to similar degrees. As hypothesized, shuffling induces the least effective cross-lingual representations. The RTD loss inhibits cross-linguality, as accuracy decreases for each layer closer to the RTD loss (L6). The "mask" model exhibits an unexpected behaviour, where its accuracy drops in its middle layers before rising up again in its output layer. We do not have a satisfying explanation for this and we leave it for future work. However, adding an MLM loss over the encoder completely changes this behaviour, and both "mask+eMLM" and "replace+tied" yield the best results.

We also notice an interesting result in the accuracy of the embeddings. Shuffling noise induces poor alignment, unlike the other methods that have much better aligned embeddings. We believe that this result is connected to the unsupervised NMT results. Note that, the quality of embeddings affect the decoder as well, as their embeddings are tied.

5 Analysis

Finetuning on parallel data drives models towards similar destinations (BLEU), but the results in unsupervised NMT hinted that their starting points are different, which implies that the finetuning process itself is critical. We shift our focus to the pretrained models themselves and using a series of probes we study how pretraining methods affect their behaviour and the knowledge that they encode. For clarity, here we discuss only the en↔de results. The en↔ne and en↔si results are included in Appendix C and are consistent with this analysis.

5.1 Encoder Denoising Capabilities

In this section, we study how well the encoders of the pretrained models are able to denoise the input. For each model, first, we corrupt its inputs using the corresponding noising method and then train a linear classifier over its encoder outputs, using the identity of the original input token as the label, similar to Brunner et al. (2019). In Figure 4 we report the perplexity (PPL ↓) of each classifier evaluated on the wmt18 en↔de devset. We report separately the scores for real and corrupted tokens. We observe that, as expected, in all models the PPL over non-corrupted tokens is almost perfect (PPL $\approx$ 1), indicating that the encoders perfectly preserve the input in their outputs. However, the results vary significantly for corrupted tokens.

[6]We follow Libovický et al. (2020) and obtain the encoder sentence representations for each layer with mean pooling, followed by zero-centering per language (separately). Then, for each source sentence we retrieve its nearest neighbor from the target sentences based on cosine distance.

Method	PPL (original)	PPL (corrupted)
mask=35%	1.2	256.5
mask=35% +eMLM	1.3	91.8
shuffle=5	4.0	1274.9
replace=35%	1.4	1816.4
replace=35% +RTD	1.9	1236.2
replace=35% +tied	1.4	1233.4

Figure 4: Perplexity (PPL ↓) of the token prediction probe for the en↔de pretrained models.

Method	Entropy (original)	Entropy (corrupted)
mask=35%	0.00	2.99
mask=35% +eMLM	0.00	3.00
shuffle=5	0.09	0.16
replace=35%	0.63	2.85
replace=35% +RTD	0.68	2.93
replace=35% +tied	0.65	2.76

Figure 5: Entropy of decoder's distributions during the reconstruction of original and corrupted tokens.

Masking The representations of masked tokens yield the lowest PPL, which implies that token reconstruction happens partially in the encoder. However, the eMLM loss over the encoder makes the outputs much more predictive of the original tokens. This shows that the reconstruction loss does not push the encoder to denoise the input well enough, unlike the similar[7] but explicit signal from eMLM.

Shuffling The PPL for original tokens is very low, whereas for shuffled tokens extremely high. This shows that the encoder does *not* fix the word order but simply relays the input to the output. This is in line with Xu et al. (2021) who showed that in Transformer-based NMT word-reordering happens in the decoder, instead of the encoder.

Replacement We observe a huge gap in how predictive the representations of real and replaced tokens are. We suspect that the encoder is "misled" by the replacements and relays their information to its outputs. Both tying and RTD help the encoder to generate representations that are more predictive of the true input. RTD, however, interferes slightly with the representations of real tokens.

5.2 Decoder Uncertainty

Next, we focus on the decoder and study how its token-level uncertainty varies while it reconstructs original and corrupted tokens. For each model, first, we corrupt its inputs using the corresponding noising method and then measure the entropy of the decoder's distributions for each target token. Low entropy values indicate that the decoder predicts the target tokens with certainty, by exploiting the encoder representations. Figure 5 shows the average entropy for original and corrupted tokens.

Masking When the decoder is presented with masked inputs, it directly copies the unmasked tokens (exactly *zero* entropy). By contrast, predicting masked tokens is naturally harder, and the decoder becomes very uncertain. Adding eMLM over the encoder does not change this behaviour.

[7]The eMLM and reconstruction losses are similar, but are applied in different places (encoder vs. decoder). The signal from reconstruction reaches the encoder through the decoder.

Shuffling The decoder of "shuffle=5" predicts all tokens with extreme certainty. Note that, in every step, the decoder has to choose the correct token out of N input tokens, instead of the full vocabulary, and combined with the constraints imposed by the ground-truth prefix, it is very easy for the model to find which input word to copy next. Therefore, we hypothesize that during pretraining the model mainly relies on the LM capabilities of the decoder.

Replacement We observe that the decoder is uncertain not only for fake but to a small extent, for real words as well. If a replacement is coherent and complies with the grammar of a given language, the decoder can be misled, which makes it "question" the identity of all words. RTD or tying with the generator show no clear effects.

5.3 Decoder Sensitivity to Encoder Outputs

This analysis aims to estimate the reliance of the decoder on the outputs (i.e., representations) of the encoder. First, we feed to the encoder a corrupted sentence $\boldsymbol{x} = \langle x_1, x'_2, x'_3, \ldots, x_N \rangle$, where x'_i denotes a corrupted token, and obtain its outputs $\boldsymbol{h} = \langle h_1, h'_2, h'_3, \ldots, h_N \rangle$. Then, we block the information of h'_i and measure how much it affects the reconstruction loss. We consider two blocking methods: (1) *zeroing*, in which we replace h'_i with a zero vector, and (2) *mixing*, in which we replace h'_i with random representations from other sentences in a batch. The amount by which the reconstruction loss increases implies how useful is the information in h'_i, and how sensitive the decoder is to them.

In Figure 6 we report the differences with and without blocking, in terms of the reconstruction loss (NLL). The models trained with shuffling originally yield the best reconstruction, which shows that it is comparatively the easiest noise for the decoder. Replacement is slightly harder than masking noise, because with masked inputs the decoder can easily tell when to copy and when to predict, while

Method	NLL (original)	NLL (zeroed)	NLL (mixed)
mask=35%	1.46	2.69	3.35
mask=35% +eMLM	1.42	3.37	4.13
shuffle=5	0.68	8.40	10.36
replace=35%	1.55	1.49	1.50
replace=35% +RTD	1.62	1.51	1.59
replace=35% +tied	1.53	1.53	1.50

Figure 6: Change in reconstruction loss (NLL ↓) after blocking the representations of corrupting tokens.

replacements can be misleading (recall §5.2).

Masking "mask" and "mask+eMLM" reconstruct the input equally well, but when the representations of masked tokens are zeroed, "mask+eMLM" is affected more. eMLM forces the reconstruction to partially happen in the encoder (recall §5.1), so the decoder relies more on it. Mixing increases NLL even more, as we inject misleading information.

Replacement RTD leads to worse reconstruction error, which suggests that it is interfering even in the pretraining phase. Surprisingly, blocking the representations of replaced words not only does not increase the reconstruction loss but even slightly decreases it. This implies that during pretraining, the models learn to *ignore* the replaced words.

Shuffling Zeroing the representations of misplaced tokens is destructive and replacing them with random representations, increases the loss even further. The decoder focuses so heavily on putting words in the right order and has no "doubts" about their identity. Therefore, when presented with missing or misleading information, instead of "falling back" into an unconditional LM, that uses only on the target prefix, it completely fails[8].

5.4 Visualization of Encoder Representations

In Figure 7, we visualize the encoder token representations using t-SNE (van der Maaten and Hinton, 2008) and color code them based on whether the belong to original or corrupted tokens (see Appendix C.1 for details and more visualizations). Masking induces separated representation between masked and unmasked tokens, which enables the decoder to copy with certainty (see §5.2), while eMLM, that pushes the encoder to reconstruct the corrupted tokens, makes them more similar to the original ones (see §5.1). Although the representations of original and reordered tokens have a large overlap, we observe some separated clusters of original and misplaced tokens, implying that the encoder is partially aware of shuffling noise. As expected, adding RTD over the "replace" models enables allows it to identify more corrupted tokens, as shown by the size of the corresponding clusters.

[8]NLL after zeroing is 8.4. It equals to PPL of $\exp(8.4) = 4447$, which is very high even for a weak unconditional LM.

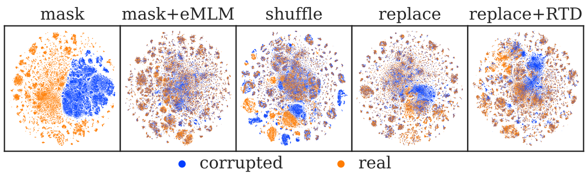


Figure 7: Visualization of encoder representations

6 Conclusions

In this work, we explore new unsupervised pretraining methods for NMT. We consider alternative objectives to masking, such as reordering or replacing input words, that produce training examples similar to real sentences.

We discover that (semi-)supervised NMT is not very sensitive to the pretraining objective. While some methods are better than others, most models converge to similar BLEU scores (§4.1, §4.2). Surprisingly, even pretraining with shuffled inputs yields competitive results with the other methods. Our ablation experiments (§4.4) imply that pretraining benefits more the decoder than the encoder.

In unsupervised NMT, however, the results vary significantly (§4.3). Shuffling noise leads to significantly worse performance, whereas masking noise, unexpectedly, yields the highest BLEU. Experiments on parallel sentence retrieval (§4.5) show that different objectives do affect the encoder cross-lingual abilities, and are reflected on the unsupervised NMT results. Through further and extensive analysis of pretrained models (§5), we find that they encode and use information in different ways.

We conclude that finetuning to each downstream NMT task is sensitive to different properties of pretrained models. (Semi-) Supervised NMT benefits from strong and fluent decoders, because the signal from the parallel data compensates for encoders with poor cross-lingual representations. Unsupervised NMT finetuning, however, requires models with good source-target mappings, and is also sensitive to certain model biases, such as the tendency to copy (§5.2) induced by mask-based pretraining.

Acknowledgments This work was conducted within the EU project Gourmet, under grant agreement No 825299. It was also supported by the UK EPSRC grant EP/S001271/1 (MTStretch). It was performed using computing resources in CSD3 using funding from the UK EPSRC (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Mikel Artetxe, Gorka Labaka, Noe Casas, and Eneko Agirre. 2020. Do all roads lead to rome? understanding the role of initialization in iterative back-translation. *Knowledge-Based Systems*, 206:106401.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the conference on machine translation (WMT). In *Proceedings of the Conference on Machine Translation*, pages 272–303, Belgium, Brussels.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2019. On identifiability in transformers. In *International Conference on Learning Representations*.

Franck Burlot and François Yvon. 2018. Using monolingual data in neural machine translation: a systematic study. In *Proceedings of the Conference on Machine Translation*, pages 144–155, Brussels, Belgium.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 7057–7067. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for Low-Resource neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 567–573, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. On the strengths of cross-attention in pretrained transformers for machine translation. *ArXiv*, abs/2104.08771.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 6097–6110, Hong Kong, China. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Leo Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Aizhan Imankulova, Takayuki Sato, and Mamoru Komachi. 2017. Improving low-resource neural machine translation with filtered pseudo-parallel corpus. In *Proceedings of the Workshop on Asian Translation (WAT2017)*, pages 70–78, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 66–71, Brussels, Belgium.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020a. Pre-training via paraphrasing. *Proceedings of the Advances in Neural Information Processing Systems*, 33.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jindřich Libovický, Rudolf Rosa, and Alexander Fraser. 2020. On the language neutrality of pre-trained multilingual representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1663–1674, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Conference on Machine Translation*, pages 186–191, Brussels, Belgium.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163, Valencia, Spain.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8(0):842–866.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 86–96, Berlin, Germany.

Aditya Siddhant, Melvin Johnson, Henry Tsai, Naveen Ari, Jason Riesa, Ankur Bapna, Orhan Firat, and Karthik Raman. 2020. Evaluating the cross-lingual effectiveness of massively multilingual neural machine translation. In *AAAI*, pages 8854–8861.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, J. Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *ArXiv*, abs/2104.06644.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 5926–5936, Long Beach, California, USA. PMLR.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Jörg Tiedemann. 2020. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Yiren Wang, Chengxiang Zhai, and Hany Hassan. 2020. Multi-task learning for multilingual neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1022–1034, Online. Association for Computational Linguistics.

Hongfei Xu, Josef van Genabith, Qiuhui Liu, and Deyi Xiong. 2021. Probing word translations in the transformer and trading decoder for encoder layers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–85, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5753–5763.

A Model Configuration & Training

We use 6 Transformer layers in both the encoder and the decoder, with embedding/hidden size of 512, feed-forward filter size of 2048, 8 attention heads and we apply 0.1 dropout to all layers. We optimize our models using Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_1 = 0.999$, and $\epsilon = 10^{-6}$. All models use sinusoidal positional embeddings.

We tie the weights of the embedding and output (projection) layers of all sub-networks (Press and Wolf, 2017; Inan et al., 2017), which involves the encoder, decoder and MLM generator. For pretraining, we use a learning rate of $5e^{-4}$ with a linear warm-up of 16K steps, followed by inverted squared decay. We train each model for 300K steps with mini-batches of 24K tokens on 8 Nvidia V100 GPUs, which requires approximately 4-5 days. The maximum sentence length is set to 256 tokens. For the finetuning experiments, we use a learning rate of $3e^{-5}$ with a linear warm-up of 2.5K steps and mini-batches of 12K tokens. We finetune each model for 60K in the supervised setting and 120K steps in the semi-supervised setting. We also use increased the dropout to 0.3 and set label smoothing (Szegedy et al., 2016) to 0.1, to avoid over-fitting on the limited parallel data.

B Unsupervised NMT

Instead of adding input noise, like Artetxe et al. (2018); Lample et al. (2018), we follow the finetuning process of mBART (Liu et al., 2020). To prevent models from copying the source during backtranslation and force the transition to the translation task, we allow only the most frequent tokens in the target languageto be generated for the first 2K steps. Specifically, we mask tokens with frequency less than 10^{-3}, as measure in the monolingual data. For model selection, we use a small subset of 200 sentences from the wmt18 devset.

B.1 Supervised Translation Ablations

In Figure 8, we visualize the experimental protocol for our ablation experiment. To test (i.e., ablate) a component, the process is the following:

1. We transfer all the weights of a pretrained model to a downstream model, except the weights of the ablated component.

we assume a shared multi-lingual vocabulary

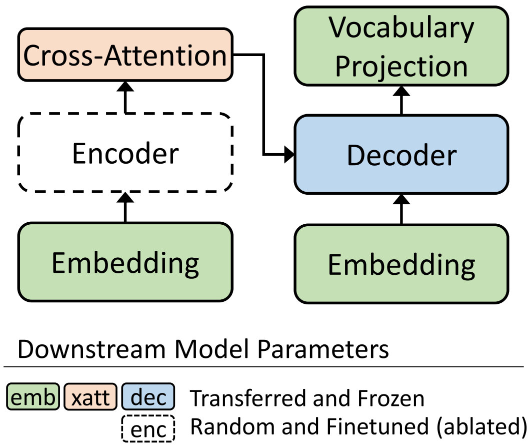
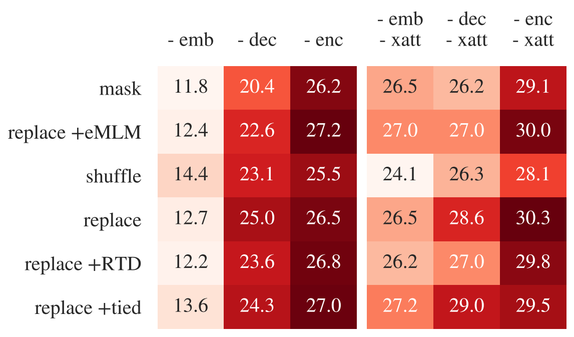

Cross-Attention
Vocabulary Projection
Encoder
Decoder
Embedding
Embedding
Downstream Model Parameters
emb xatt dec Transferred and Frozen
enc Random and Finetuned (ablated)

Figure 8: Visualization of the ablation experiment, using the ablation of the encoder as example. The figure shows the high-level architecture of a model and the colors correspond to the parameter set. The input embeddings of the encoder, the decoder and the vocabulary projection have the same color as they share the same parameters. During finetuning, we update only the weights of the ablated component, and all the other (pretrained) weights are frozen.



	- emb	- dec	- enc	- emb - xatt	- dec - xatt	- enc - xatt
mask	11.8	20.4	26.2	26.5	26.2	29.1
replace +eMLM	12.4	22.6	27.2	27.0	27.0	30.0
shuffle	14.4	23.1	25.5	24.1	26.3	28.1
replace	12.7	25.0	26.5	26.5	28.6	30.3
replace +RTD	12.2	23.6	26.8	26.2	27.0	29.8
replace +tied	13.6	24.3	27.0	27.2	29.0	29.5

Figure 9: Ablation results for supervised NMT on en→de (wmt19). We reset each main component individually (left) and with the cross-attention (right).

2. We freeze the pretrained weights and finetune only the ablated (i.e., randomly initialized) component.

We decided to follow this protocol, in order to isolate the effects on the final BLEU score on the ablated component, and to also prevent the other components from compensating. In concurrent work, Gheini et al. (2021) have considered a similar experimental protocol, but to study a different but related phenomenon. In Figure 8, we show the ablation results for the en→de direction.

C Analysis Results in Other Languages

In Figures 10, 11, 12, 13, 14, 15, we report the analysis results 5 on the en↔ne and ↔ne pretrained models, evaluated on their NMT dev sets.

We observe that the results are very consistent with those for en↔de.

Training Details of Probing Classifier For the analysis in Sec. 5.1, each classifier is trained on the monolingual data for 50K steps, and optimized with Adam using a learning rate of 0.0001. Only the parameters of the classifier are updated and the rest of the model remains *fixed*.

Method	PPL (original)	PPL (corrupted)
mask=35%	1.0	474.0
mask=35% +eMLM	1.1	109.7
mask-replace=35%	1.0	285.2
shuffle=5	3.0	723.2
replace=35%	1.3	1906.6
replace=35% +RTD	1.4	1596.5
replace=35% +tied	1.3	1269.1

Figure 10: Perplexity (PPL↓) of the token prediction probe for the en↔ne pretrained models.

Method	PPL (original)	PPL (corrupted)
mask=35%	1.0	408.1
mask=35% +eMLM	1.1	169.9
mask-replace=35%	1.0	356.4
shuffle=5	2.9	680.5
replace=35%	1.3	2026.4
replace=35% +RTD	1.5	1629.1
replace=35% +tied	1.3	1419.1

Figure 11: Perplexity (PPL↓) of the token prediction probe for the en↔si pretrained models.

Method	Entropy (original)	Entropy (corrupted)
mask=35%	0.00	3.14
mask=35% +eMLM	0.00	3.23
shuffle=5	0.14	0.26
replace=35%	0.66	2.95
replace=35% +RTD	0.68	3.09
replace=35% +tied	0.72	2.89

Figure 12: Decoder entropy for the reconstruction of real/corrupted tokens, for the en↔si pretrained models.

Method	Entropy (original)	Entropy (corrupted)
mask=35%	0.00	3.51
mask=35% +eMLM	0.00	3.55
shuffle=5	0.19	0.39
replace=35%	0.83	3.21
replace=35% +RTD	0.86	3.30
replace=35% +tied	0.84	3.02

Figure 13: Decoder entropy for the reconstruction of real/corrupted tokens, for the en↔ne pretrained models.

Method	NLL (original)	NLL (zeroed)	NLL (mixed)
mask=35%	1.67	2.46	3.03
mask=35% +eMLM	1.69	4.16	4.09
shuffle=5	1.25	9.95	10.12
replace=35%	1.76	1.75	1.71
replace=35% +RTD	1.84	1.78	1.83
replace=35% +tied	1.74	1.60	1.63

Figure 14: Reconstruction loss (NLL ↓) with/without blocking the outputs of corrupted tokens (en↔si).

Method	NLL (original)	NLL (zeroed)	NLL (mixed)
mask=35%	1.93	3.12	3.60
mask=35% +eMLM	2.00	3.76	4.22
shuffle=5	1.45	9.80	10.12
replace=35%	2.08	2.01	1.97
replace=35% +RTD	2.17	2.05	2.09
replace=35% +tied	2.07	1.95	1.93

Figure 15: Reconstruction loss (NLL ↓) with/without blocking the outputs of corrupted tokens (en↔ne).

C.1 Visualization of Encoder Representations

In this section, we visually inspect the input token representations. We compare each method based on how the representations evolve through the layers of the encoder, by focusing on two aspects of each token, (1) its *language* and (2) whether it has been *corrupted* or not. The goal is to inspect how each model encodes these two types of information about the input tokens.

Methodology We sample 5K sentences from the English-German monolingual data and pass them through the encoder of each model using corresponding noising method. For each token, we keep its representations from every layer and label them by language and noise. We keep only the representations of the 2K most frequent tokens and exclude the representations of special tokens, such as the `[BOS]`, `[EOS]`, and language IDs, which significantly skew the results. The final dataset contains approximately 100K token representations per layer (600K in total). For the visualization, we project to 2D with t-SNE (van der Maaten and Hinton, 2008). For each model, we visualize the representations of its encoder per layer (L1 to L6), which we colour-code by language (top-row) and the identity (bottom-row) of each token. L1 refers to the outputs of the first Transformer layer, therefore the tokens have been contextualized once.

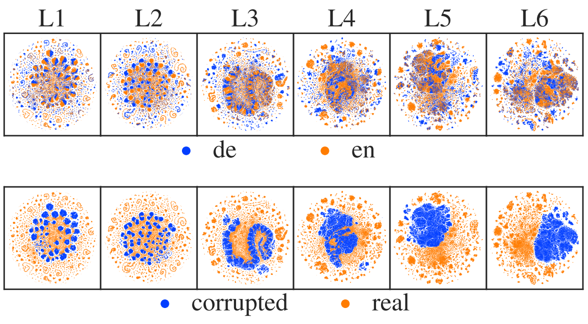


Figure 16: Visualization of encoder representations from the "mask=35%" model.

Masking In Fig. 16 we visualize the encoder of the "mask=35%" pretrained model. Real and masked tokens occupy different regions, whereas the tokens from each language are much closer to each other. All models trained with masking noise exhibit similar behaviour. In the first layer, the masked tokens are organized into multiple small clusters, but the encoder progressively groups them into larger structures. We visually verify that the encoder keeps the masked tokens separated even in the last layer. This aligns with our findings in Sec. 5.1, which suggest that the reconstruction loss does not incentivize the encoder to denoise the representations of masked tokens. Also, it enables the model to easily identify the real tokens and can copy them, as discussed in Sec. 5.2.

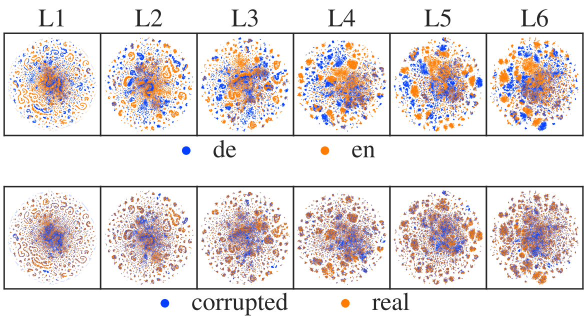


Figure 17: Visualization of encoder representations from the "mask=35%+eMLM" model.

Masking+eMLM In Fig. 17 we visualize the "mask=35%+MLM" model. Adding eMLM makes the masked and real tokens are indistinguishable from each other. Intuitively, to minimize the MLM loss the encoder must generate representations that are predictive of the true identity of the masked tokens, therefore similar to real tokens. We observe a small overlap between English and German tokens, as there are more language-specific clusters. We believe that is because the eMLM loss pushes the representation to better encode the grammar and semantics of each token.

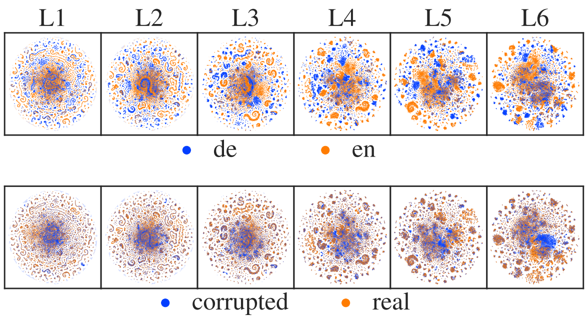


Figure 18: Layer-wise visualization of encoder representations from the "replace=35%" model.

Replacement In Fig. 18 we visualize the "replace=35%" model. There is a moderate separation between languages, slightly less than the "mask=35%+MLM" model. However, we observe that the representations of real and fake tokens generally overlap with each other, unlike the "mask=35%" model, especially in the lower layers. This is because the model is always given as input embeddings of actual words and not `[MASK]`, which unless contextualized all of them are treated the same. Only in the last layer, we can see the formation of a fake-only cluster. This suggests that the pretraining objective (i.e., reconstruction) creates a bias towards discriminating between real and fake token. However, the separation is not as extreme as in the masked-based models but is not obvious if the encoder can't or is not biased to more clearly separate them.

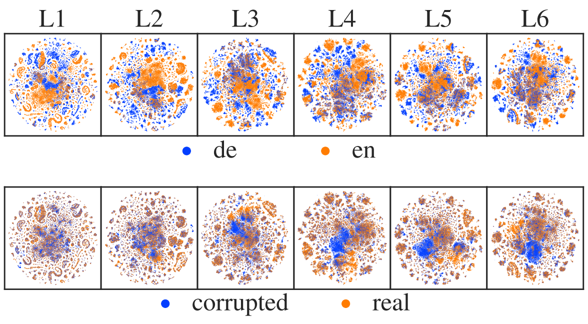


Figure 19: Visualization of representations from the "replace=35%+RTD=4" model, that is trained with an RTD head over the 4th layer of the encoder.

Replacement Token Detection In Fig. 19 we visualize the "replace=35%+RTD=4" model, that is trained with an RTD head over the 4th layer of the encoder. The only difference with the "replace=35%+RTD=6" is that the effects of RTD start to show earlier. Compared to the "replace=35" the

Method	**en→de**		**de→en**		**en→ne**	**ne→en**	**en→si**	**si→en**
	wmt18	wmt19	wmt18	wmt19				
random	26.2±0.1	25.3±0.1	27.6±0.1	19.1±0.3	3.3±0.1	6.5±0.1	2.5±0.1	6.5±0.1
mask=35%	33.3±0.1	30.7±0.2	33.2±0.0	**25.4**±0.0	5.1±0.1	10.2±0.1	3.7±0.0	10.0±0.1
mask=35% +eMLM	33.4±0.0	30.6±0.1	33.5±0.1	25.2±0.2	**5.3**±0.0	**10.8**±0.1	**4.0**±0.0	10.4±0.1
mask=35% (span)	33.3±0.1	30.5±0.1	33.4±0.0	25.2±0.0	5.1±0.1	10.1±0.1	3.9±0.1	9.9±0.1
shuffle=5	31.6±0.1	28.7±0.0	31.7±0.0	23.9±0.1	4.9±0.0	9.9±0.1	3.4±0.0	10.1±0.1
replace=35%	33.9±0.0	30.3±0.2	33.5±0.1	**25.4**±0.1	5.1±0.1	9.9±0.1	3.7±0.0	9.8±0.0
replace=35% +RTD	32.9±0.1	30.0±0.0	32.5±0.0	24.4±0.1	5.0±0.0	9.9±0.1	3.4±0.1	9.7±0.2
replace=35% +tied	**34.2**±0.0	**30.8**±0.1	**33.7**±0.1	25.3±0.2	**5.3**±0.0	10.6±0.1	3.7±0.0	**10.5**±0.1
+ shuffle=3	34.0±0.0	$\underline{31.1}$±0.1	33.4±0.1	25.1±0.2	$\underline{5.5}$±0.0	$\underline{11.0}$±0.0	4.0±0.0	$\underline{10.8}$±0.1
+ dec: mask=15%	33.9±0.1	$\underline{30.9}$±0.0	33.6±0.1	25.3±0.2	$\underline{5.5}$±0.0	10.5±0.0	3.9±0.0	10.4±0.0
+ dec: replace=15%	$\underline{34.5}$±0.1	30.7±0.1	33.4±0.0	$\underline{25.6}$±0.1	$\underline{5.6}$±0.0	10.5±0.1	3.9±0.0	$\underline{10.7}$±0.1

Table 5: Finetuning results to supervised NMT. "dec:X" denotes method that add noise to the the decoder. We report the average of 3 runs and the standard error of the mean.

representations are more clustered and less spread-out, even in the lower layers. The real and fake tokens are much better separated, and the separation peaks at layer 4, which visually verifies the bias introduced by RTD, but the separation is not as extreme as for masking noise. Although this suggests that separating masked/original words is harder than real/fake, it also depends on the weight used for the RTD loss during pretraining, which is not something we have explored. Also, the visualization suggests that the separation remains approximately constant in the remaining layers. There is no perceptible difference with "replace=35" in terms of language.

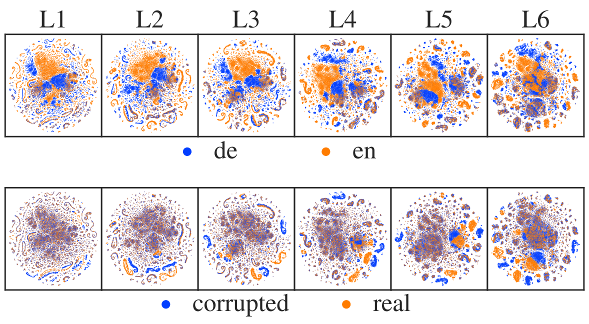


Figure 20: Visualization of representations from the "shuffle=5" model.

Shuffling In Fig. 20 we visualize the "shuffle=5" model. We observe a strong separation between languages, that slightly decreases in the upper layers. The language clusters are relatively large, unlike the much smaller and local language-specific clusters seen in the other models. As for noise, initially all tokens are represented similarly, but as the encoder re-contextualizes the input, it puts more misplaced tokens in separate clusters. This shows that the model becomes progressively more "aware" about the misplaced tokens in the input. After manual inspection of the clusters with corrupted tokens, we found that the majority of them contain punctuation marks. This makes intuitive sense, as it should be easy for the model to identify punctuation marks that has been misplaced.

D Additional NMT Experiments

In Table 5 we report some additional results for supervised NMT 4.1 omitted from the main paper. Besides combining shuffling and word replacements in the input, we also introduce noise in the decoder side, by randomly replacing 15% words of the decoder's input with the `[MASK]` token ("dec:mask=15%") or with samples from the generator ("dec:replace=15%"). Note that, we reduce the amount of noise in this case to avoid disrupting training. Overall, we observe that both methods increase performance in some language pairs, but the improvements are marginal.

We also experimented with RTD over shuffled inputs, by training the model to explicitly detect if words were misplaced or not, but this configuration lead to poor results.

Bowman et al. (2016) reported that masking more than 20% of words in the decoder hurts its LM capabilities.